1987

Design for a prototype Marine Corps officer staffing model.

Hayes, Mark E.

Monterey, California: U.S. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

DESIGN FOR A PROTOTYPE
MARINE CORPS OFFICER STAFFING MODEL

by

Mark E. Hayes

December 1987

Thesis Advisor: Gordon H. Bradley

REPORT DOCUMENTATION PAGE

REPORT SECURITY CLASSIFICATION: Unclassified	1b RESTRICTIVE MARKINGS:
SECURITY CLASSIFICATION AUTHORITY:	3 DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; Distribution is unlimited.
DECLASSIFICATION/DOWNGRADING SCHEDULE:	
PERFORMING ORGANIZATION REPORT NUMBER(S):	5 MONITORING ORGANIZATION REPORT NUMBER(S):
NAME OF PERFORMING ORGANIZATION: Naval Postgraduate School; 6b OFFICE SYMBOL (If applicable): Code 52	7a NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
ADDRESS (City, State, and ZIP Code): Monterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code): Monterey, California 93943-5000
NAME OF FUNDING/SPONSORING ORGANIZATION; 8b OFFICE SYMBOL (If applicable):	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER:
ADDRESS (City, State, and ZIP Code):	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO; PROJECT NO; TASK NO; WORK UNIT ACCESSION NO

TITLE (Include Security Classification)
DESIGN FOR A PROTOTYPE MARINE CORPS OFFICER STAFFING MODEL (u)

PERSONAL AUTHOR(S)
Hayes, Mark E.

TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month Day)	15 PAGE COUNT
Master's Thesis	FROM _____ TO _____	1987 December	102

SUPPLEMENTARY NOTATION

COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Personnel; allocation, assignment, design, model

ABSTRACT (Continue on reverse if necessary and identify by block number)

This thesis is a design of an officer allocation model for the United States Marine Corps. It is a prototype software system that supports the modeling of various officer assignment policies. Officer allocation is the mapping of officer types to classes of billets. Assignment is the matching of a particular officer to a specific billet. This thesis examines the current officer allocation capability and proposes the design of a software system that will permit those making assignments the capability to define the type of allocation model that will best support their needs.

DISTRIBUTION/AVAILABILITY OF ABSTRACT: ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION: Unclassified
NAME OF RESPONSIBLE INDIVIDUAL: Prof. Gordon H. Bradley	22b TELEPHONE (Include Area Code): (408) 646-2359; 22c OFFICE SYMBOL: Code 52Bz

Approved for public release; distribution is unlimited.

Design for a Prototype Marine Corps Officer Staffing Model

by

Mark E. Hayes
Captain, United States Marine Corps
B.S., U. S. Naval Academy, 1980

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 1987

ABSTRACT

This thesis is a design of an officer allocation model for the United States Marine Corps. It is a prototype software system that supports the modeling of various officer assignment policies. Officer allocation is the mapping of officer types to classes of billets. Assignment is the matching of a particular officer to a specific billet. This thesis examines the current officer allocation capability and proposes the design of a software system that will permit those making assignments the capability to define the type of allocation model that will best support their needs.

THESIS DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to thank Professor Bradley for all of his support and patience in working with me. I would also like to thank all of my friends that encouraged me, fed me, and prayed for me as I worked on the thesis.

I. INTRODUCTION

A. BACKGROUND

The primary mission of the U.S. Marine Corps is to defend the U.S. and its interests. In order to accomplish this mission, the Marine Corps established a personnel structure that organizes people into various sized combat and support units. There are approximately 190,000 marines in the Marine Corps, of which 20,000 are commissioned officers and 170,000 are enlisted. The Marine Corps has three major combat units known as Marine Amphibious Forces (MAF). There is one each on the East and West coasts, and one split between Hawaii and Japan. There is also a fourth MAF that is make up of marine reservists. Each MAF consists of a combat Division, a Force Service Support Group, and an aircraft Wing. Each of these units can be broken down again several times into their most elemental units. There are also marines stationed at bases, schools, and other posts that support the Marine Corps. Each of the different units require people with special and specific skills in order for each unit to effectively support the Marine Corps missions. Within the officer corps there are 168 such skills or Military Occupational Specialties (MOS). Each officer has a designated primary specialty and may have up to two additional specialties designated as additional MOS's (AMOS's). In addition, each officer has a designated rank or grade such as Lieutenant, Captain, Major, etc.

Each unit in the Marine Corps has a Table of Organization (T/O) that specifies how many officers (and enlisted) are required of each MOS and grade to fully man the unit. These T/O's are maintained at Headquarters Marine Corps (HQMC) in Washington, DC. The Manpower Management Officer Assignment Branch (MMOA) at HQMC is tasked with ensuring that all of the authorized officer billets in the Marine Corps are filled. There is also a similar branch that handles the enlisted, however, it will not be discussed. MMOA has two sections of officers known as monitors. Each monitor is responsible for billets with different sets of MOS's and grades. For example, one monitor may handle Artillery and Financial Management Captains and Lieutenants, while another monitor may take care of Artillery and Infantry Majors. The monitors keep track of each of their authorized officer billets and assign officers to fill those billets. Historically, about seven to eight thousand officers are assigned to

new jobs (militarily known as billets) each year. Each billet can be specifically described by the command (unit) it is at, a MOS and a grade. In making assignments the monitors consider special command needs, career patterns of the officers, staffing requirements, preferences of the individual, the population of officers, the number of officers in each MOS, the relative priority of billets and the staffing policies in effect at the time. Because all of these factors do not all agree completely, it is not always possible for the monitor to assign an officer that exactly matches the given requirements for every billet. How well an officer matches the requirements of a billet is known as the 'fit'. There are also several billets, usually at support commands, that are not restricted to just one MOS or grade. These billets, then, may be filled by one of several monitors. The filling of these billets must be coordinated between those monitors.

To aid the monitors in making their assignments, the Officer Staffing Goal Model (OSGM), a computer-based model, was developed and implemented over ten years ago. The purpose of the OSGM is to provide each monitor with an optimal staffing goal target; that is, a staffing goal with the greatest possible number of billets filled and the best possible fit for those officers. A staffing goal is a listing of each billet at each command and the description of a type of officer that can fill that billet. The OSGM does not attempt to consider special command needs, career patterns and individual Marine preferences because these change very frequently and are very difficult to model on a computer. Its output is therefore considered a target for the monitors to strive for and is used as a tool by them when making their assignments. Again, the OSGM has two objectives when creating staffing goals: to fill the largest number of billets possible and to have the best 'fit' for each billet filled. It is not normally feasible to fill all the billets that are authorized or to have the best 'fit' for every billet filled. The current OSGM takes care of this by first getting the best fill possible and then within that constraint finding the best fit attainable.

There are three inputs to the OSGM: the inventory of the current Marine Corps officer population, the listing of all authorized billets (known as the ASR), and the 'Dictionary'--a collection of data used by the OSGM to apply current Marine Corps assignment policy to the model. Using these inputs, the OSGM performs several functions in developing the staffing goal.

The first of these functions is the updating of the ASR. The ASR is generated three times a year. The ASR is a subset of the T/O. Again, the T/O is a listing of all

the personnel needed to fully complement a unit. But because of budget and manpower constraints, not every unit can be fully manned. Therefore, the ASR was established to record which billets out of the T/O are authorized. At each model run, all of the changes to the billet structures that have been approved by HQMC in the previous four months are applied to the ASR. Since the OSGM may be run more often than this and should be as up to date as possible in order to give the best possible solution, the ASR can be adjusted by the OSGM to reflect any changes in the authorized officer strengths that have occurred since the ASR file was generated, or when the authorized strength is not broken out in a manner needed by the model. The adjustments to the ASR can be increases or decreases to current authorizations, setting authorizations to specific levels, or adding new authorizations that previously did not exist. An example of the ASR is given in Figure 1.1 .

BMOS	GRADE	MCC	NO. of BILLETS
0802	03	V15	9

Figure 1.1 Authorized Strength Report (ASR).

In a similar manner the Inventory or officer population must also be kept up to date. The Inventory is reduced to reflect those first term officers who will be leaving active duty. These reductions are accomplished by randomly pulling officers of specified grades and MOS's from the Inventory before the model is run. The grades and MOS's can either be specified by major categories or by specific grade and MOS. The officer inventory can also be adjusted to show accessions or attritions for other than first term officers. Any officer type (specified by MOS, grade, experience, sex and additional MOS) can be increased, decreased or set to a given level. An example of the Inventory data is shown in Figure 1.2 .

The Dictionary is used to specify what type of officer can fill a given billet type at a command (this is known as the substitutions for a billet). Each officer is described by his primary MOS, any additional MOS's, his grade and indicators that show the officer's sex, experience level and LDO/unrestricted status. Each billet has a set of

PMOS	GRADE	AMOS	AMOS	MCC	EXP	UR	SEX
0802	03	0402	0000	MC4	1	1	M

Figure 1.2 Officer Inventory.

similar descriptions that specify the requirements for the officer filling the billet. The relative priority in which billets should be filled is also specified. Using the information from the Dictionary, the adjusted Inventory and the adjusted ASR, the OSGM then finds the best fill and fit possible. The method in which the current OSGM derives the staffing goal will not be discussed.

B. CURRENT ISSUES

There are many issues today that impact the Marine Corps and the way in which it assigns its officers to billets. One of the most obvious to the military itself is the rising cost of PCS (Permanent Change of Station) moves. In this type of move the officer and his/her family are moved from one duty station to another. Increased cost and Congressional limits on PCS funds have impacted all of the services. One such impact is the delay of orders that have already been issued. Another impact of increased PCS costs is the possiblity of increased tour lengths. Current Marine Corps stateside tours are 36 months long. These could be increased to 48 months in order to decrease the number of yearly moves. Another issue now facing all of the services is that of joint tours. A joint tour is where a member of one Armed Service is stationed with another of the Armed Services. It is intended that this type of tour will improve inter-service operability. These billets must be established in each of the services with compensatory reductions elsewhere in each branch. These billets must also be handled by the monitors and therefore included in the OSGM. With increased public awareness of the cost of the military has come an increased interest on the part of Congress in the internal workings of each of the services (in some circles this is known as micromanagement). Because of this, the Marine Corps must be able to, even more than before, judge the impact of any proposed personnel policies before they are implemented. A much dreaded thing in each of the services is manpower cuts. There

have already been some and will probably be more. The Marine Corps must be able to gauge the impact of these cuts in order to see what changes to make in order to maintain operational readiness. Finally, the current OSGM has essentially reached its maximum capability. It does not have the ability at this time to adequately handle any of the issues just mentioned. It could be improved, but the expense would far outweigh the benefits.

C. PURPOSE

This thesis will provide MMOA-3 with a prototype design for an allocation model. The design is not a smooth, production design because it does not attempt to provide a final, fixed form of solution. Instead, it gives the user, MMOA-3, a flexible solver system that can be modified with relative ease. This will allow the users to develop the specifications for a new system with great confidence that the new system, based on the modified prototype model, will satisfy their requirements.

D. THESIS ORGANIZATION

The thesis will be organized in the following manner. The user requirements will be defined in the terms used by the user. A statistical analysis will be presented with any impacts the results have on any of the user requirements. The proposed design will be detailed both at the system level and at the detailed design level. Following the design will be a discussion of some of the major requirements and how the design answers those requirements. Next, the prototype used to validate the design will be presented and an analysis of the results of the prototype given. Finally, the conclusions and recommendations derived during the development of the model will be discussed. A glossary of terms is provided in Appendix A.

II. USER REQUIREMENTS

The user requirements for the prototype were developed over a period of several months. The problem was initially introduced during a week long visit to MMOA-3 at HQMC in Washington, DC. Each of the staff at MMOA-3 was interviewed, as well as some of the monitors and personnel department heads. An initial problem statement was drafted and then presented to Major Hundley of MMOA-3 for review while at HQMC. During the following three months, the problem statement was further refined and again reviewed by Major Hundley three more times. The resulting user requirements are presented below.

A. COMPUTER

One of the first requirements of the prototype model is that it be capable of being run on the Marine Corps mainframe at Quantico, VA. Currently, the OSGM is maintained on a contractor's mainframe in Maryland. A detailed procedure must be followed to transfer all of the working data files to the contractor's mainframe so that a model run may be done. Since MMOA has terminals to the Quantico mainframe, having the prototype on the mainframe would allow easy access for running or modifying the prototype. Also, the cost for maintaining the system would be much less.

B. WINDOW SIZE

Another requirement for the prototype model is that it have a variable 'window' size. One of the preparations that MMOA makes when it wants to run the OSGM is to produce the latest inventory of officers. This is a listing of all Marine Corps officers that have at least one year of active duty service remaining. The date on which the program that produces the inventory of officers is run is called the staffing goal date. The 'window' is a period of time, beginning on the staffing goal date. Any officers whose current tours of duty end any time within the window are eligible to move and are called 'movers'. The movers are the officers that will be assigned to new billets by the monitors. Any officers whose tours of duty end before or after the window are not eligible to move and are called 'non-movers'. In the current OSGM the window is

fixed at one year. However, the desire is to have a variable window size. For example, the window could be as small as one month or as large as three or four years. If the window were one month, the set of movers would be very small, while the set of nonmovers would be very large. If the window were over three years, essentially the entire officer corps would be classified as movers.

C. COST FACTORS

A third requirement for the prototype design is that it incorporate various 'cost factors'. In the current OSGM, there are two measures of effectiveness used to gauge the performance of a model run. The first is the number of billets that are filled by the model. Obviously, if the model fills a low percentage of billets, the results will not be very useful. Even though the measure is expressed in the number of people, it is usually referred to as a percentage of the officers allocated. The other measure of effectiveness is the average, over the entire population of officers, of how well each officer matches or 'fits' the billet he is allocated to. As mentioned previously, in a normal run not all of the billets can be filled and the degree of fit is usually not the maximum. There must, then, be a tradeoff between the two. Another factor that can be used to gauge the performance of a model run is a total PCS cost. This factor is not supported on the current OSGM. It is, however, an important piece of information when considering budget cuts and congressional inquiries. The total PCS cost for a model run is calculated by summing up the estimated PCS cost for each mover. This estimate is based on the mover's current command location and the location of his/her allocated command that he/she may move to. The estimate may be continually refined by updating the data from (recent) historical records. The PCS cost may just 'tag along' as an extra piece of data in the model solution and be summed up after the solution or it may be used as a criteria by the solver routine when matching officers to billets. A listing of the cost factors and their descriptions are given in Figure 2.1 .

D. CRITERIA

There are currently seven criteria used to describe each officer in the inventory and each billet listed in the ASR. The criteria are: primary MOS (PMOS), grade, two additional MOS's (AMOS), an experience marker, a use restriction marker that indicates whether the person/billet is an unrestricted officer/billet or an Limited Duty Officer (LDO) officer/billet, and a sex marker. Figure 2.2 shows the criteria with

FACTOR	DESCRIPTION
Fill	Actual number of billets filled
Fit	1 : Best fit 5 : Worst fit
PCS	S0 : Nonmover S0 - Max : Mover

Figure 2.1 Cost Factors.

examples of each. All of these must have a valid value and are used in matching the officers to billets. The final requirement is that the prototype design facilitate new criteria being added and used in the matching process, also that the design allow the optional removal of one or more criteria from consideration for matching on any given model run. An example of a new criteria that might be incorporated in the model is a joint MOS (JMOS) designator given to officers who qualify for joint tours and to billets that require an officer with such a designator. It is also possible for experience, use restriction or sex markers to be selectively removed from consideration so that the impact they play on the matching of officers to billets could be studied. The same type of logic could be applied to any new criteria that are added at a later date.

CRITERIA	EXAMPLE
PMOS	0802 - Artillery, 0302 - Infantry
Grade	O2 - Lieutenant, O3 - Captain
AMOS	9646 - Computer Science, 0402 - Logistics
Use Restriction	1 - LDO, 2 - Unrestricted
Experience	0 - No experience, 1 - Experience
Sex	M - Male, F - Female

Figure 2.2 Matching Criteria.

E. MISCELLANEOUS

Although no formal time constraints were given for the model run time, MMOA desired that the prototype be as efficient as possible. It is anticipated that the prototype may be run frequently as new aspects of the allocation problem are modeled using the prototype. A desire (that was not given as a requirement) was that the prototype use the data currently used by the OSGM as input, in the same format, so that duplicate forms of data would not have to be maintained.

III. DATA ANALYSIS

A. SCOPE

The statistical analysis presented here is a simple study of the personnel and billet data, intended to discover the character and properties of the data. It is not intended to be a detailed, complex analysis checking for correlations between data elements, etc. The knowledge derived through this analysis enhances the understanding of the problem and permits greater insight into possible solutions.

B. ANALYSIS

The analysis will cover several different data sets, all of which are directly used in the allocation problem. First, a breakdown of different attributes of the inventory of officers will be shown, then the division of officers among grades. The various sources of billets will be given, followed by a listing of the number of billets assigned to each staffing precedence level (SPL). The next two tables show the breakdown of the E1 and E2 'cards' from the dictionary. Finally, the number of officers in each of the officer type categories will be presented.

1. Inventory

Within the inventory of officers, several different groupings occur. The ones shown in Table 1 are by MOS, use restriction, experience and sex. Just over half of the officers in the Marine Corps have only a PMOS. Fewer still have one AMOS and two AMOS's. An analysis presented later will show how many types of billets require no, one, or two AMOS's. This is important because those officers with one or two AMOS's are qualified to fill billets that do not ask for an AMOS, and if pulled to fill such billets may leave other, more specialized billets vacant. The Limited Duty Officers constitute a small percentage of the officer corps and should, therefore, represent a small part of the entire allocation problem. Over half of the officers are designated to be experienced. Whether or not an officer is considered to be experienced depends on his/her rank, number of years in their MOS, and other factors. Finally, the male/female split is given.

2. Grade

As can be seen in Table 2, the greatest bulk (over half) of the officer corps are Lieutenants and Captains. This follows directly from the pyramid-shaped organization

TABLE 1
INVENTORY

Total Officers on Active Duty	20050	
Officers with PMOS only	11543	57.5%
Officers with PMOS, one AMOS	4986	24.9
Officers with PMOS, two AMOS	3521	17.6
Unrestricted Officers	18785	93.7%
Limited Duty Officers	1265	6.3
Officers with Experience	11290	56.3%
Officers with no Experience	8760	43.7
Male Officers	19392	96.7%
Female Officers	658	3.3%

of the Marine Corps. The Warrant Officers serve in specialized billets, for the most part, and are all prior enlisted. The general officers are not presented in this data because they are not included in the allocation model. They are 'monitored' by the Commandant of the Marine Corps.

3. **Billet Source**

Not all of the billets filled by officers are listed in the Authorized Strength Report (ASR). As seen in Table 3, there are over three thousand training billets. The majority of these are student billets at Marine and other service commands. For example, approximately nine hundred of the billets are for Second Lieutenants at the Basic School in Quantico, VA. Others are MOS training billets at various schools.

4. **Billet Description**

The figures in Table 4 are derived from the data produced in a partial allocation model run. As seen in the next table, the lowest numbered Staffing Precedence Level (SPL) has the highest priority. The SPL is a descriptor given to every command or unit that indicates its relative priority for getting its billets filled.

TABLE 2
OFFICER GRADES

	No. of Officers	
Warrant Officer	1628	8.1%
Lieutenant	6351	31.7
Captain	6258	31.2
Major	3293	16.4
LT Colonel	1713	9.5
Colonel	807	4.1

TABLE 3
BILLETS

Authorized Strength Report	16708
	No. of Billets
Training	3152
Nonchargeable	721

The figures shown below are a function of the data given to the model and can, therefore, be modified. One factor that greatly affects the method of solution, but is not specified anywhere in the data itself, is the fact that the billets of each SPL are filled preemptively. In other words, all of the billets of SPL 2 that can be possibly be filled are filled first. The same is done for SPL's 3, 4, and 5, in turn.

5. **Command Priority**

Table 5 gives the breakdown of the number of E2 'cards' by SPL. The term 'card' is a holdover from the time that the data for each E2 record was actually on a punch card. The E2 records have two main functions. The first is to assign each billet or set of billets in the ASR an SPL. This can be seen in the previous table where the billets are divided by SPL. The second function is to tie each billet listed in the ASR to a set of billet descriptors (E1 cards). These are explained in the next paragraph.

TABLE 4
SPL BILLETS

SPL 2	1149	6.9%
SPL 3	6649	40.2
SPL 4	1776	10.8
SPL 5	6973	42.1
Total	16547	100.0

TABLE 5
E2 SPL

SPL 1 (Highest Priority)	1	0.02%
SPL 2	504	12.8
SPL 3	2567	65.1
SPL 4	84	2.1
SPL 5 (Lowest Priority)	782	19.8

As just mentioned, each billet is tied to a set of billet descriptors. These are the E1 cards. As seen in Table 6, there are up to five choices for each billet. There is a logical progression of fewer cards for each greater numbered choice. Those billets that have very stringent requirements will have fewer choices than those that will take a wide range of officers.

The officer type is a descriptor used in the E1 cards. It is used for billets that may be filled by an officer from a set of MOS's. This type of billet requires a general, rather than specific, background. Each of the officer types listed includes all grades, Warrant Officer through Colonel. As fitting a combat service, the ground and air combat officer types are the largest. Even if the figures given were reduced to reflect only a single grade, it is easily seen that any billet that specifies one or more officer types will have tens or hundreds of officers that are qualified to fill that billet. This

TABLE 6
E1 PRIORITY

PRI 1 (1st Choice)	1019	46.3%
PRI 2	597	27.2
PRI 3	370	16.8
PRI 4	174	7.9
PRI 5 (5th Choice)	40	1.8

presents a special problem for the solution designed in this thesis, as will be seen later. A breakdown of the officer types given in Table 7 .

TABLE 7
OFFICER TYPES

Air/Ground Combat Support	558	3.9%
Air/Ground Service Support	555	3.9
Ground Combat	4813	33.7
Ground Combat Support	1172	8.2
Ground Combat Service Support	2806	19.6
Naval Aviator Fixed Wing	1726	12.1
Naval Aviator Helicopter	2277	15.9
Naval Aviator NFO	387	2.7
Total	14294	100.0

The data presented in this last table, Table 8, is a breakdown of the billet descriptor (E1) cards. The E1's have many 'wildcards' that allow many options for the various descriptors. As can be seen for experience, use restriction, and sex, the largest proportion of billet descriptors are those that allow either. For MOS's, the single largest category is that of the PMOS only. Relatively few billet descriptors require a PMOS and one or two AMOS. There are also billets that do not specify what PMOS

an officer must have, only that they carry a certain AMOS. There are also over four hundred billet types that ask for any officer within a large range of PMOS's. The Occupational Field and SubOccupational Field are similar to the Officer Type, but on a smaller scale. These ask for an officer in a particular MOS area; for example, an officer in the finance/disbursing MOS field.

TABLE 8
BILLET DESCRIPTORS

No Experience Required	571
Experience Required	518
Either Experienced or Inexperienced	1706
Unrestricted Officer Required	260
LDO Required	38
Either LDO or Unrestricted	2497
Male Officer Required	40
Female Officer Required	8
Either Male or Female	2747
PMOS Only Required	1388
PMOS and 1 AMOS	481
PMOS and 2 AMOS	8
AMOS and no PMOS	334
Officer Type	415
Occ/Sub-Occ Field	150

IV. SYSTEM DESIGN

A. DESIGN PRINCIPLES

Through conversations with MMOA about their requirements and through the study of the data used in the allocation model, several design principles were developed. These principles were used in the development of this design. Please note that the order in which the principles are presented reflects no relative priority among them. The first principle developed is that the model should support iteration. The model will not be used just occasionally, but may be used repeatedly throughout each month. The design should allow the user to repeatedly use the model, entering the model each time at the point most appropriate for the work being done. The second principle is driven by the relatively large data sets and the amount of computation necessary during a model run. Any operation should be done on the smallest data set possible. This will reduce both the model run time and the model run cost. The third design principle is that the design should be modularized, supporting easier maintenance, and future expansion or changes. It is much easier to modify a small module, rather than a large, monolithic one. There is a trade-off here in that the larger number of modules requires that some data be written to and read from files in order to pass the data from module to module. For the purposes of this prototype, however, the cost is acceptable. the last design principle is one that is hard to completely specify and that is flexibility. The word is overused and often misunderstood. In this particular instance, flexibility means the capability to use different officer and billet descriptors for matching purposes on various model runs with only a few minor code modifications. This includes the ability to add new, currently unknown descriptors to the model in the future. The entire design presented here emphasizes the functions of each module. The module interdependencies are simple and straightforward. Block diagrams of each level of the design and the system design are given in Figures 4.1 through 4.3, and Figure 4.4, respectively.

B. HIGH LEVEL

The high level design provides the model the ability to be selectively iterated. There are three major modules in the design. The first is the initial inventory preparation. All inter-module communications are accomplished using files--data files.

The second is the data preparation and matching module. The third and final is the solution and report generation module. The modules are presented in the order of execution. The initial inventory preparation module converts and compresses necessary data into formats usable by the prototype and eliminates all data no longer needed by the model. This reduces the amount of unnecessary processing. It also generates some reference data files that are used by the other modules. The data preparation and matching module allows the user to specify which officer/billet descriptors will be used in matching officers to billets, and then performs the matching. Once the first module has been run, the second module may be run and rerun many times, without having to reexecute the first module again. The output of the second module is a listing of all possible matches of officers to billets. The solution and report generation module takes this data, prepares it for the solver, solves the allocation problem and then generates the requested reports.

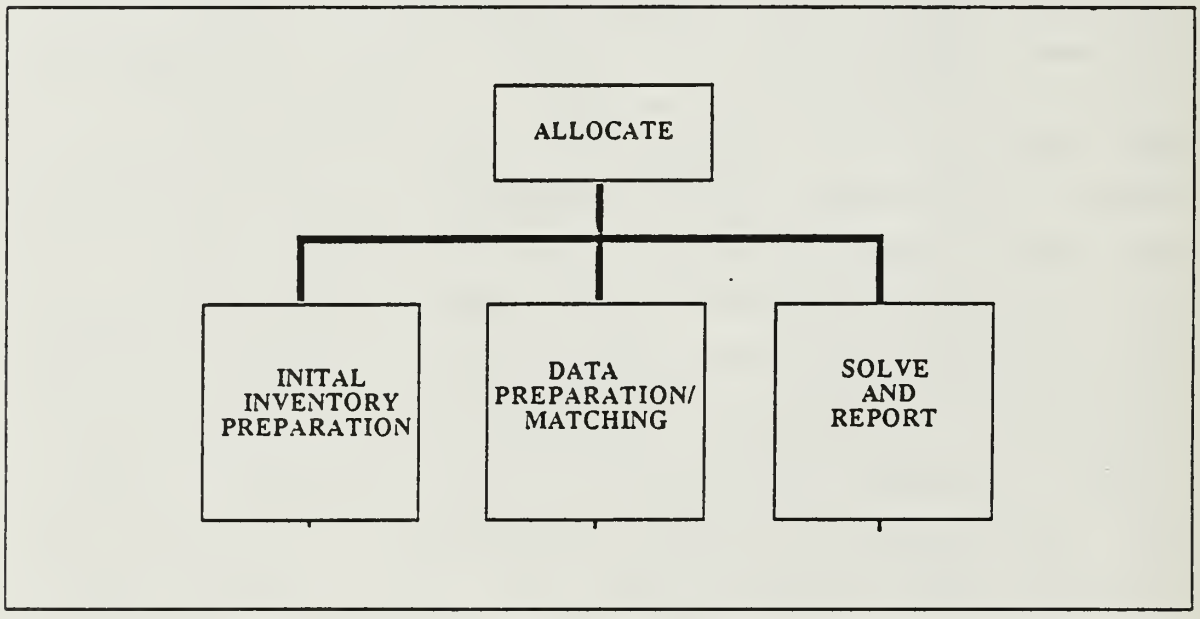

Figure 4.1 High Level Design.

C. MID LEVEL

The mid level design implements the modularization. The initial inventory preparation module is made up of two submodules. The first, using the staffing goal date and window provided by the user, splits the inventory of officers into movers and

nonmovers. Because of the nature of the data, the nonmovers are not set aside as might be imagined. In order to improve the quality of the solution, the nonmovers are matched to billets just like the movers are with one exception. The nonmovers are eligible to be matched only to billets in the command they are currently serving in. After the inventory is split, the second submodule removes any officers currently in training or nonchargeable billets from the inventory. This is because the primary function of the prototype is to fill the ASR billets. The mover inventory and the reduced nonmover inventory is now available to be used by the second major module.

The data preparation and matching module is made up of three submodules. The first submodule prompts the user for which officer/billet descriptors he wishes to use. It presents the criteria currently used for use or discard. It then allows the user to specify new criteria that have been added to the model. It then creates a file with indicators for each of these descriptors. This is an optional module in that it need not be executed every time the prototype is run. It is only run when the criteria used for matching are to be changed. It applies these descriptor indicators to the inventories in preparation for the matching. The second submodule uses the same file and applies the same indicators to the billet data. The third submodule takes the inventories from the first submodule and the billet data from the second submodule and merges the two, matching each officer to every billet he/she is qualified to fill.

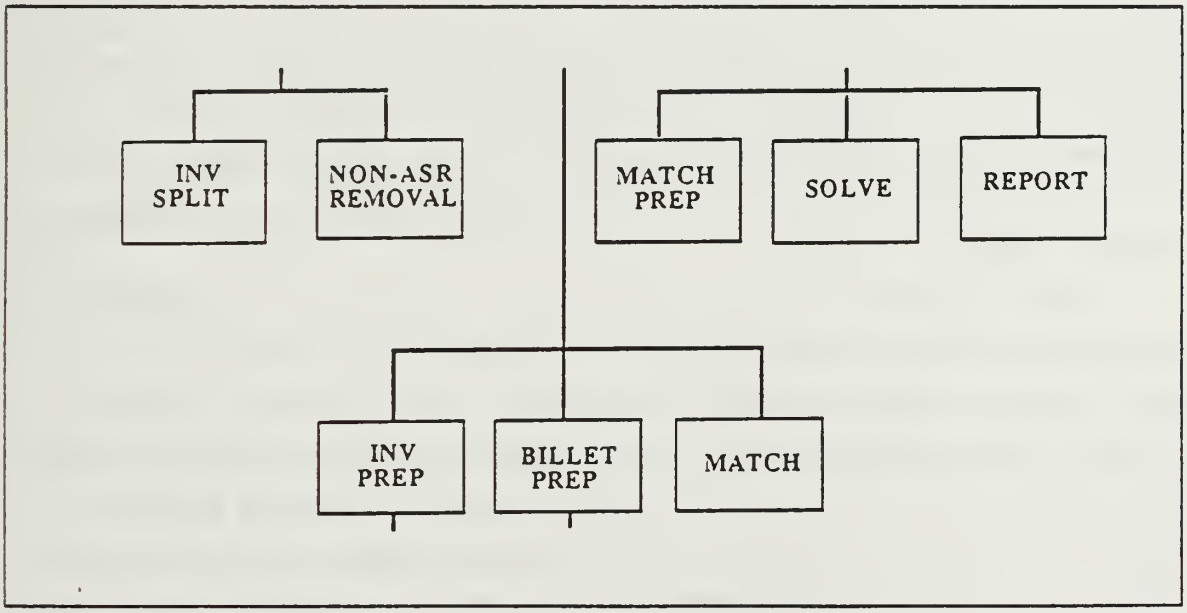

Figure 4.2 Mid Level Design.

The third major module contains two submodules. The first takes the file of matched officers and billets, puts it in a form that can be processed by the solver, uses the data from the match file to generate costs that will be for the solver and then generates an optimal solution. The second module is the report module that generates reports from the data produced by the solver.

D. LOW LEVEL

There are four low level modules. Two of the four support the inventory preparation submodule and the other two support the billet preparation submodule. The first module applies the grouping criteria decided upon by the user to the inventory of 'mover' officers. There are currently three standard criteria that the user can use to group the officers and billets. They are the experience, use restriction and sex of the officer or billet. If only these three were ever used , then the logic could be 'hardwired'. However, the user requirements specify that the design be able to facilitate these three, or more, or fewer. This is done by allowing any number of criteria to be used (within reason!). The combinations of criteria are then determined. All of this data is then abstracted into an ordered set of indices. This is the key to the flexibility. When the matching module is making matches, it is checking a criteria number and not an officer's experience, sex, etc. This design allows as few criteria as one, or as many as the user desires. At the same time, the module also classifies each officer as to how many AMOS's he/she has. This classification becomes very useful in the matching routine because it eliminates much unnecessary checking of officers that may have the correct criteria index but do not have the correct number of MOS's. The PMOS and/or AMOS's must still be checked to see if they match the billet descriptor, but no needless checking is done. The same procedure is applied to the inventory of 'nonmover' officers.

A similar principle to the inventory grouping is applied to the grouping of the billet descriptors. However, it is a little more complicated because the billet descriptors usually allow one additional choice for each criteria. That extra choice is an 'either'. In other words, a billet may require a male officer, require a female officer or allow either a male or female to fill the billet. The billet descriptor grouping is, therefore, a superset of the inventory grouping. To connect the two, a mapping is created from the inventory grouping into the billet descriptor grouping. This mapping is used in the matching module.

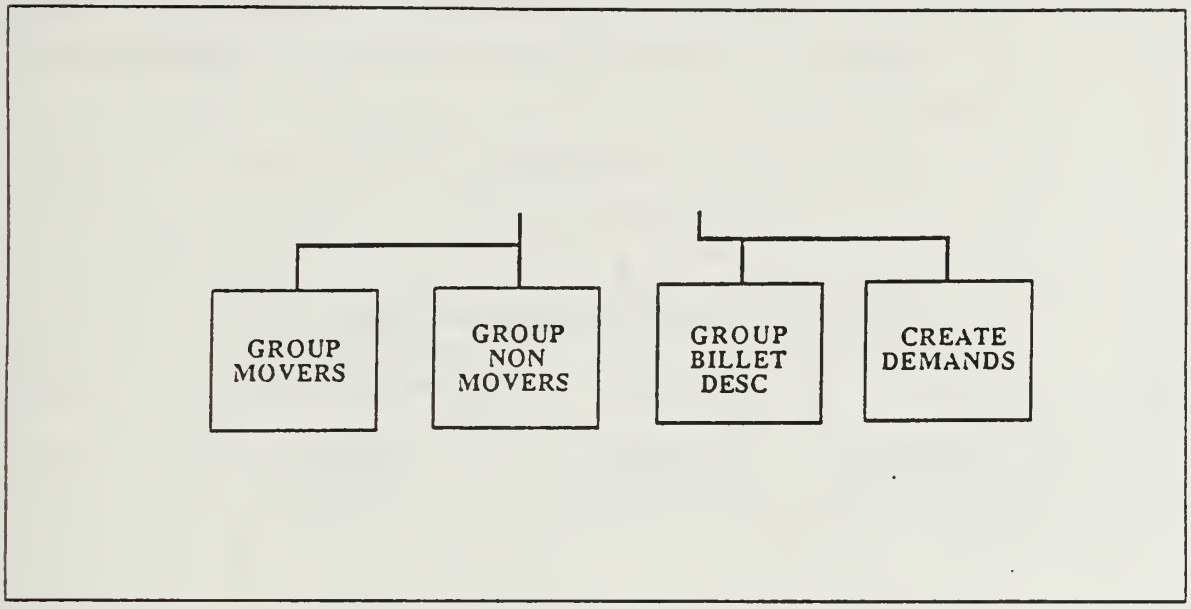

Figure 4.3 Low Level Design.

The final low level module is the module that generates the demands. A demand is a record that contains one billet descriptor, one priority level, one staffing precedence level (SPL), and one command code (MCC). Since every distinct billet type can have up to five billet descriptors, priorities one to five, each billet type can have up to five demands. There is one caveat to this. A single billet descriptor may specify more than one grade. If it does so, when the billet descriptor is expanded to one grade per line, the billet will have more than five demands. Although there are more demands than there are officers, this presents no problem. The demands are used to generate potential matches. The solver looks at all the potential matches and then selects only as many as are needed to fill the billets.

E. SOLVER

The design and function of the solver drives the design of the rest of the system. This is because the solver is the core of the system. All of the processing prior to the model is done to prepare the data for the solver. The report generator takes the results of the solver as its input. The model developed for this system is a derivative of a network assignment model developed by Capt. Phil Exner, USMC [Ref. 1]. Capt. Exner designed the model as the centerpiece of his thesis. His model is a multiobjective optimization built on a capacitated transshipment network model and is an application of work done by D. Klingman [Refs. 2,3] and R. Rosenthal [Ref. 4]. The solver is a

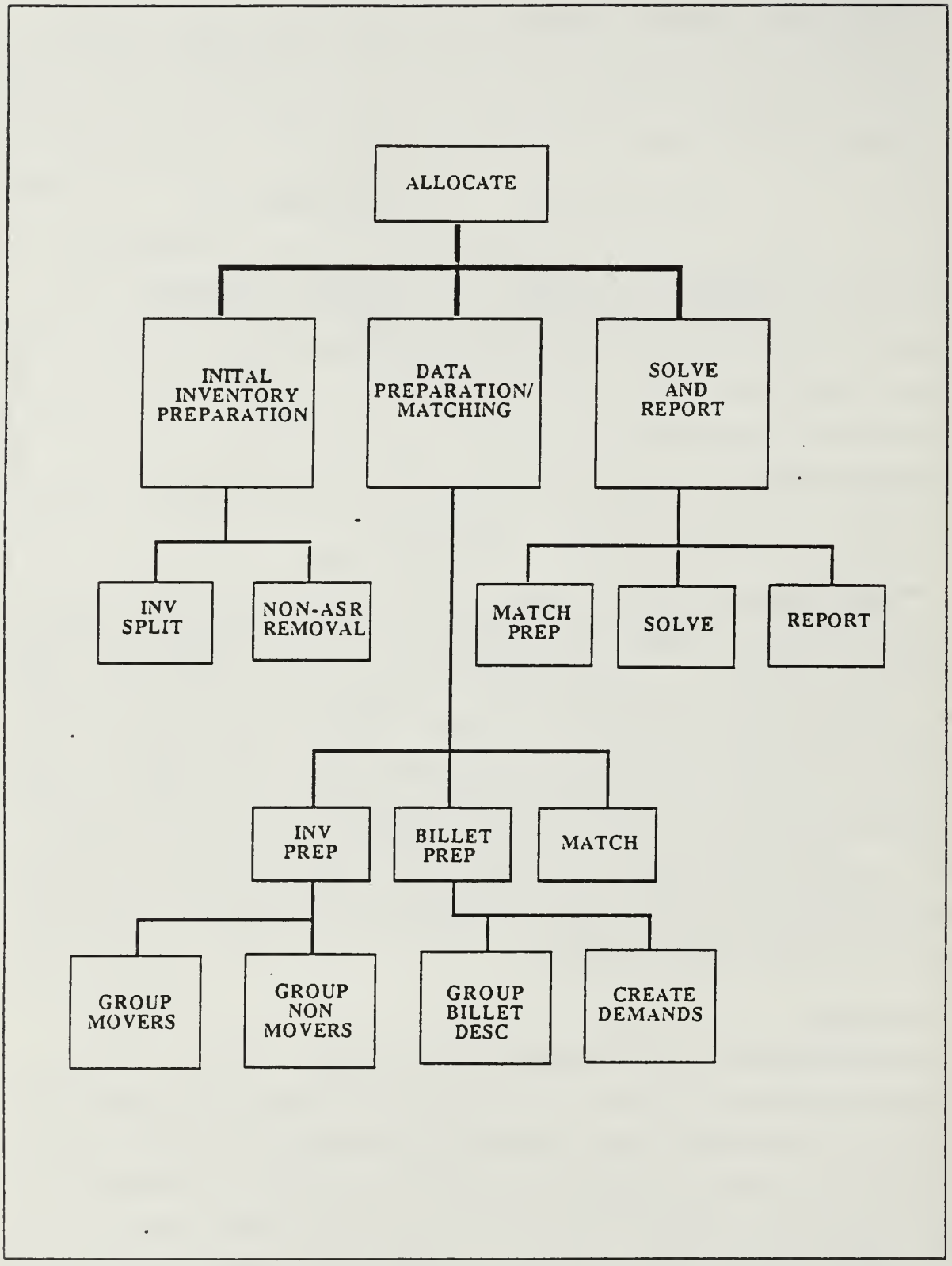

Figure 4.4 System Design.

variant of GNET [Ref. 5]. The mathematical basis for the model will not be discussed. If further information in this area is desired, please see the references.

The concept of the solver, despite all of its complicated background, is fairly simple. It essentially matches producers of a commodity to users of the same commodity. The diagram in Figure 4.5 helps to illustrate. The commodity for the model is people - officers. Each person is a producer or source. Each billet is a 'user' of an officer, needing one officer to fill it. In the diagram, the officers are listed on the left. Each officer has a specific set of descriptors (PMOS, grade, experience, etc.). The billets are listed on the right. In this case, each circle is not just one billet, but may represent several billets (BMIS), each of which is at the same command and requires the same set of officer attributes. The officer type circles (or 'nodes' as they are called in the network) are both users and sources. The act as channels to funnel hundreds of officers whose PMOS is in an officer type to the billets that request officers by officer type. Again, each officer is connected to every billet he/she is qualified for. Each 'connection' is known as an 'arc' in the network. Each arc has an associated cost. Normally this is a combination of the fit and PCS costs. Obviously, the officer with the lowest fit number (the best fit) and the lowest PCS cost would be the best choice for a billet. The solver takes all these nodes and arcs as data, along with the associated costs as input. It then fills as many billets as possible, with the lowest aggregate cost possible. The result is a one-to-one mapping, with each officer being allocated to only one billet, and each billet having only one officer. It is very possible, even likely, however, that some billets will not be filled and some officers not allocated. This is not the fault of the solver but is an artifact of the problem and data. It should also be noted that if the model is run using officer types the PCS costs will only be an estimate because any officers that are matched to a billet via an officer type have no direct PCS cost basis and can only use an average PCS cost. However, if the model is run without using officer types, the PCS cost will be as accurate as the costing data provided.

There is a difference between this model and the one developed by Capt. Exner. The model presented here solves the problem in a hierarchical fashion, filling as many billets as possible and then within that constraint finding a solution with the smallest aggregate cost. Because of this, solving for the smallest PCS cost will not reduce the number of billets filled. Capt. Exner's solver, on the other hand, is constructed in a way that allows the user to reduce any cost factor in order to improve the performance of another. In that case, the number of billets filled could be reduced to improve the fit or toal PCS cost.

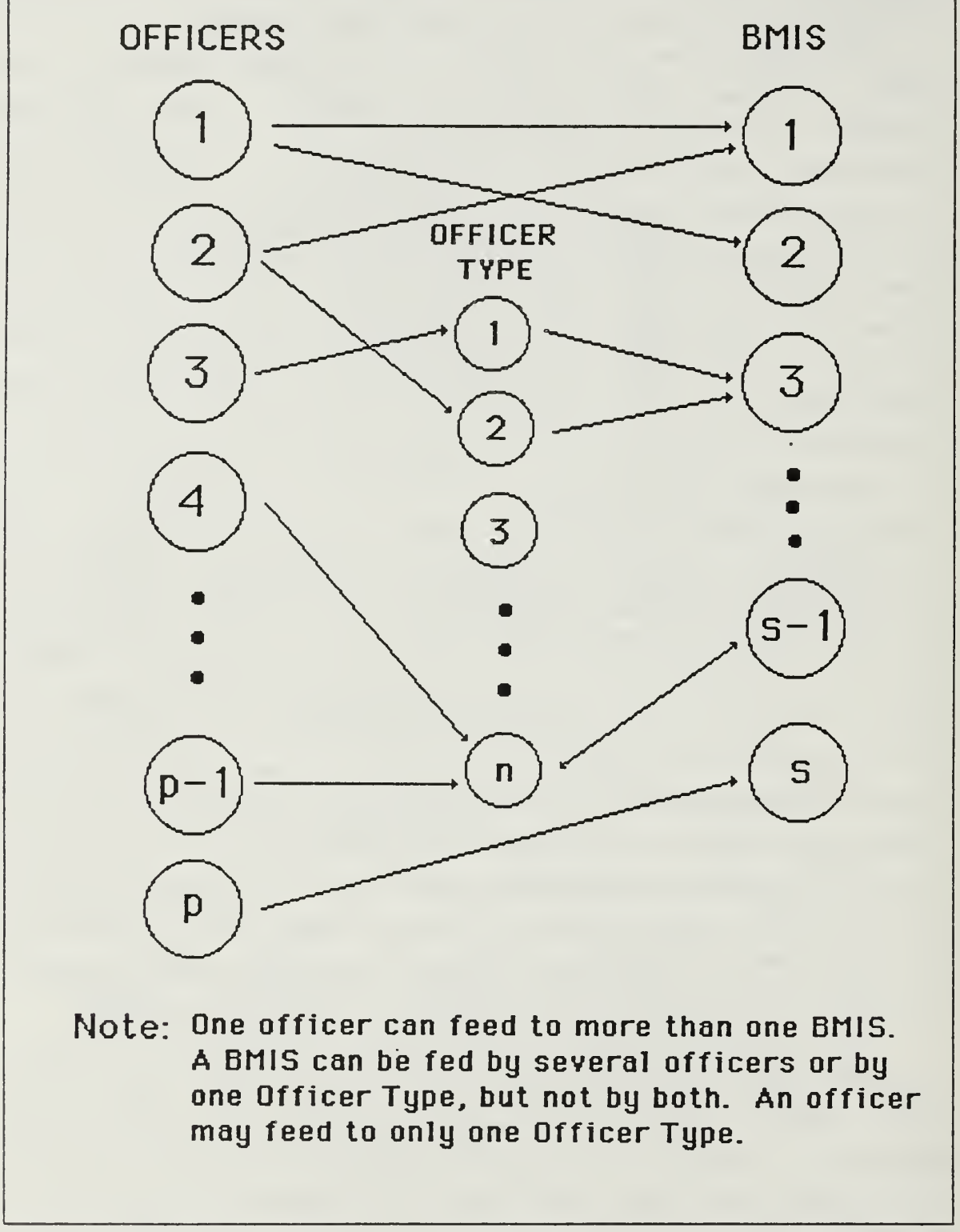

Figure 4.5 Logical Structure of the Model.

F. REQUIREMENTS AND DESIGN RESPONSE

The user requirements for the system were outlined in Chapter II. A brief description of the system design was presented in Chapter IV. The following is a demonstration of how the design met the requirements set out by the users.

1. Computer

The requirement that the prototype be capable of running on Marine Corps computer assets was easily met. A review of the software available on the mainframes at Quantico, VA and here at the Naval Postgraduate School revealed two common software packages: a Fortran-77 compiler and a SAS package. Therefore, the limited prototype developed as a part of this thesis to validate the design given in the thesis was written primarily in Fortran-77. All of the sorting routines and one or two other miscellaneous routines were written in SAS. This will be discussed in more detail in Chapter VI.

2. Window Size

As with the first requirement, the requirement for the prototype model to have a variable window size was easily met. The design includes a module in the preprocessing section that allows the user to input the staffing goal date (in YYMM form) and the window size in months. It then uses an algorithm provided in the OSGM manual [Ref. 6] for determining if an officer is a mover or not. Since the window size is adjustable and drives the relative percentages of the mover and nonmovers, the mover and nonmover data files are variable in size. This variability is data set sizes is supported throughout the entire design.

3. Cost Factors

The requirement for the design to incorporate various cost factors, some of which are unknown at this time, is accomplished in two ways. First, the calculation of the cost from the data provided is limited to a single module. Since the details of future cost factors is unknown, this limits the scope of the work necessary to incorporate the new cost algorithms. Second, the ability to 'carry' data through modules that do not affect the data is made straightforward. The logical procession of data and data transformations is easy to follow. As mentioned in Chapter Three, there are two cost factors currently used as measures of effectiveness for the OSGM: fill and fit. Fill is derived after the allocation solution has been obtained. Fit is a cost input to the solver, as is PCS cost. The design provides the framework for calculating PCS cost from the mover's current and future command locations. Any new cost factors that

can be derived directly from currently provided data can be computed in the arc generation module. New data added to the model to provide a new cost factor must be added to the inventory of officers. It may also be added to the billet descriptor and matching data if also used to determine billet eligibility. This type of change will require the most modification.

4. **Criteria**

As just mentioned above, if any new data is added as a new criteria, it must be added to the officer inventory, billet descriptors and matching criteria. The design was developed to favor the addition of simple criteria with a 'yes-no' applications such as experienced or inexperienced. This type of criteria may be added to the grouping procedure with the least amount of modification. Only the data and modules up to the grouping modules might need minor modification. All of the modules after the grouping would be unaffected.

If a new criteria can not be reasonably added in the grouping module, the new data would have to be carried through the model up to the matching modules. The methodology for matching often will be fairly uniform, allowing straightforward modifications. This type of criteria addition, obviously, is the most difficult.

Another feature provided by the design is the ability to use a subset of the grouping criteria. For example, if it is desired to determine the effect of using versus not using experience as a matching factor, it could be done. The user has the ability to choose or disallow any grouping criteria already implemented. This is done without any code modification at all. Once the user has selected his subset of grouping criteria, the model reconfigures itself as it works through the modules. This level of abstraction can be extended to cover any new criteria added to the system.

V. DISCUSSION OF PROTOTYPE

A. SCOPE

A limited prototype of the design was implemented in order to validate the design concepts and provide a working model for the user. The prototype is considered limited because it does not implement every feature of the current OSGM and does not implement every feature of the design to its fullest extent. One factor affecting this was time constraints. It was not possible in the time allotted for the thesis to develop the design and produce a full implementation of the design. However, the major factor affecting the development of the prototype was the desire of the user to have a working model that could be easily modified. In this regard, the prototype can be looked at as a structure or framework that provides the basic functions necessary to develop a solution. These basic functions can then be modified or rearranged to provide a new configuration. Listings of the programs in the prototype are given in the Appendices.

B. DISCUSSION

The prototype was developed in three parts in order to support the designed ability to selectively iterate through the model. The three parts are the initial preprocessing stage, the intermediate processing stage and the final or solution stage. The user can enter the prototype at the beginning of any of these three stages. The initial preprocessing stage is designed to be run every time a new inventory of officers is developed which is normally once a month.

1. Initial Stage

In this first stage, three MOS reference files are created. Then the officer inventory is split according to the staffing goal date and window size. The nonmover inventory is then reduced by removing those officers in the training billets listed in the Dictionary. The resulting mover, nonmover and reference files are now available to be used as often as desired.

2. Intermediate Stage

The intermediate processing stage performs the groupings and matches. It first allows the user to choose from the grouping criteria already provided by the prototype or to add new criteria whose values have been added to the inventory and billet data files. The groups are made by taking the cross product of the choices of the criteria chosen.

It can be seen in the example in Figure 5.1 that if a large number of criteria are used or several criteria with a large number of choices, the number of groups will grow exponentially and no benefit will be gained by the grouping. In fact, the prototype performance could be worsened. The larger the number of groups, the fewer the officers or billets that will be in any one group and the more processing that will be required. Therefore, a balance must be maintained. It is estimated that performance will decrease rapidly as the number of groups approach and exceed one hundred. This grouping is applied to both the movers and nonmovers. A superset of the grouping is also applied to the E1 (billet) data.

	USR	EXP	SEX
	UNR	Y	M
	LDO	N	F

GROUP	USR	EXP	SEX
1	UNR	Y	M
2	UNR	Y	F
3	UNR	N	M
4	UNR	N	F
5	LDO	Y	M
6	LDO	Y	F
7	LDO	N	M
8	LDO	N	F

Figure 5.1 Sample Grouping.

Next, the ASR is expanded, adjusted, numbered and split into BMIS sets (E2 cards). A BMIS set is a set of billets at the same command, with the same SPL and monitor, that have the same set of billet descriptors. Each BMIS number has associated with it the number of billets needed to be filled for that BMIS. The BMIS sets are then split by SPL and merged with the billet descriptors (E1 cards) to make the billet demands. Finally, the movers and nonmovers are merged with the billet demands to create the matches. The match files contain the inventory number of the officer, the BMIS number of the billet and the cost data needed to calculate the cost of the match for every match in the file.

3. Final Stage

The final or solution stage uses the match files to create the arcs needed by the GNET solver and then solves the matches for each SPL in turn. The solution can be developed in this way because the SPL's are preemptive. The problem could not be partitioned by MOS or grade as it is by SPL because the billet descriptors allow MOS and grade substitutions. This means that there are no clear cut divisions of the problem by MOS or grade.

The first SPL solved for is SPL0 which covers all of the nonchargeable billets. The nonchargeable billets are filled first so that a truly representative inventory can be used to fill the authorized billets from the ASR. After this, SPL's 1 through 5 are filled, in sequence. After each SPL, the remaining matches in the match files are reduced by removing those officers allocated to higher SPL billets. This reduces the problem size for each subsequent SPL and provides a better performance.

The other program in the final stage uses the solution developed by the GNET solver and accesses the inventory and billet files to produce reports. It needs to access the inventory and billet files because all the solver gives in the solution is the inventory number and the BMIS number. This last program provides only the essential tools for making the reports and does not attempt to generate all of the currently used reports. It would be possible to restructure this program to put the solution data in a form usable by a database package. This would allow ad hoc queries by the user. Discuss some of the specifics of the prototype and how these meet the design specifications.

C. ANALYSIS OF RESULTS

The actual and estimated performance times for the prototype implementation are given in Tables 9, 10 and 11 Using the standard one year window size, the officer population was divided into 7,251 movers and 13,815 nonmovers. It should be noted here that SAS was used only for the sorting of the data sets. Fortran was used in all of the actual data conversion and matching. Using PULLE3T FORTRAN, 427 nonmover officers that were in training billets were removed from the nonmover inventory. The remaining training billets and all of the nonchargeable billets will be filled during the SPL0 run of the solver. Operating on the two parts of the officer inventory in the intermediate processing stage produced better performance times than operating on the inventory as a whole. The majority of the intermediate stage

programs are for billet data preparation. The match files produced by the two MATCH programs were reduced in size because the officer type billet descriptors were not included in the processing. This allowed a significant time savings in the processing time. A time estimate for using the solver iteratively on the separate SPL match files produced better results than the projections solving of the entire problem as one large problem. The decision to solve the problem by SPL's was motivated by a recommendation given in Capt. Exner's thesis [Ref. 1].

TABLE 9
INITIAL PREPROCESSING

PROGRAM	SECONDS
NUMMOS FORTRAN	0.40
INVSPL FORTRAN	58.33
BSORT SAS	18.58
EXPE3 FORTRAN	0.63
E3SORT SAS	0.62
DIVE3 FORTRAN	0.87
PULLE3T FORTRAN	26.24

TABLE 10

INTERMEDIATE PROCESSING

PROGRAM	SECONDS
MKCRIT FORTRAN	0.02
GRNINV FORTRAN	28.95
GRMINV FORTRAN	15.31
E1SORT SAS	3.44
EXPE1 FORTRAN	7.51
GRE1 FORTRAN	6.17
EXPASR FORTRAN	9.17
EXPC1 SAS	0.15
ASORT SAS	8.10
CSORT SAS	0.32
ROLLC1 FORTRAN	0.51
ADJASR FORTRAN	8.62
NUMASR FORTRAN	9.46
E2SORT SAS	4.83
BMIS FORTRAN	6.72
REND FORTRAN	7.32
BMIX FORTRAN	24.83
MSORT SAS	8.88
DEMAND FORTRAN	40.93
NMATCH FORTRAN	60.00 (est)
MMATCH FORTRAN	75.00 (est)

TABLE 11

FINAL PHASE

PROGRAM	SECONDS
SHUFFLE SAS	50.00 (est)
SOLVE FORTRAN	180.00 (est)
REPGEN FORTRAN	90.00 (est)

VI. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

There were several conclusions drawn from the results of the design and from the work performed in developing and verifying the design. The first is that the design is a viable one and does provide the flexibility and expandability requested by the user. The design is sufficiently robust to allow future enhancements and modifications with relative ease. Another conclusion is that the allocation problem can be broken down into smaller sub-problems and each of them solved individually. Due to the nature of the solver, the total time of solving the smaller problems can be significantly less than the time to solve the entire problem as a whole. Along this same line, it is also true that smaller, more specialized allocation problems can be easily set up to be solved. For example, if a special scenario were established that only required a small composite group of Marines, only the officer inventory and billet descriptor data files would have to be changed in order to run the problem through the model. Another conclusion that was recognized early in the development of the design was that the solution method could be generalized for more than just military applications. Many commercial companies have similar personnel and job requirements. Any criteria that can be quantified can be used as matching criteria. This would allow the companies great flexibility in establishing job requirements. The final conclusion was that the current method of billet description can be simplified. The major problem lies in the method of PMOS specification. Most of the billets specify a single PMOS for each choice. Some, however, specify a range of allowable PMOS's by using occupational fields, suboccupational fields or officer types. Figure 6.1 gives a sample of the number of officers in an occupational field and officer type for the grades specified. Each of these officers would be listed once for every billet they are eligible for. It is not reasonable, with the type of solver used in the prototype, to list this many officers for every billet that calls for an occupational field or officer type. A much smaller set of eligible officers would still allow a good solution.

B. RECOMMENDATIONS

Based on the conclusions presented above, these are some recommendations for future improvements to the officer allocation problem as it relates to this model. The

OCC FIELD	GRADE	NR OF OFFICERS
30xx	O3	363

OT	GRADE	NR OF OFFICERS
GDCB	O2	1155

Figure 6.1 PMOS Specifications.

first is that the results generated by the prototype be evaluated and traced in order to see which of the billet division records (E2) and the billet descriptors (E1) were actually used in the development of the solution or to what degree they were used. It is possible that due to the nature of the solver used in the prototype that the billet data may be able to be condensed. For example, it may be found that some of the third, fourth, and fifth choice billet descriptors never get used in the generation of the solution. These could be removed, and reduce the processing time and the data file size. Following this same line of thought, it would be possible to simplify the billet descriptors, PMOS descriptors in particular, if a reduction of the scope of the PMOS descriptors were done. For example, the occupational field 30xx might be replaced by 307x, which has only 27 Captains (O3's). This kind of specification would allow the monitors to be more exact in billet requirements while not losing any ability to fill all of their billets because the prototype solver inherently uses a wider officer population base than the current solver. Another recommendation would be to put the results of a model run in a database, with the data divided between the monitors. This would allow the monitors to make ad hoc queries of the data with the results of the query in any form they wished. A final recommendation is that a good PCS cost basis be developed. The prototype allows the calculation and summation of PCS costs for a model run. Updating and maintaining the PCS cost factors would allow greater confidence and better planning based on the PCS costs generated during a model run. Additional information could easily be added to the personnel data file for use in the PCS cost calculation. This data would be the number of dependents in each age group that receives different moving allowances.

APPENDIX A
GLOSSARY OF TERMS

AMOS	Additional MOS - a subspecialty (MOS) that an officer may have.
ASR	Authorized Strength Report - a listing of all authorized officer billets.
BMIS	a set of one or more billets with the same command, billet descriptor, SPL, billet MOS and billet grade.
BMOS	Billet MOS - recommended MOS of the officer who will fill the specified billet.
DICTIONARY	a collection of files used by the model in modifying the Inventory, ASR and making the allocations for the staffing goals.
EXCESS	a nonmover who cannot fill any ASR billet within his MCC.
FILL	number of billets, for a model run, that have personnel allocated to them.
FIT	how well the officer type allocated to a billet matches the stated requirements for that billet.
FIXED	a person whose tour does not end in the window and therefore is not considered for a billet outside of his MCC for the model run. The same as NONMOVER.
FREE	a person whose tour ends in the window. The person is free to be allocated to any billet at any MCC that he qualifies for. The same as MOVER.
GRADE	the rank of an officer.
INVENTORY	a current listing of the Marine Corps officer population.
MAC	Monitor Activity Code - identifies the monitor who is responsible for the data associated with the MAC.
MCC	Monitored Command Code - a code that specifies the command under which a Marine is serving.
MMOA	Manpower Management Officer Assignment Branch - the office in HQMC responsible for assigning officers to billets.
MONITOR	officer in MMOA responsible for filling all the authorized billets for a given set of MOS's and grades.
MOS	Military Occupational Specialty - Ex: Infantry, Artillery, Tanks, etc.

NON-CHARGEABLE	a person who is either a prisoner, patient, transient or in training (P2T2).
OFFICER TYPE	1) A group of officers with the same PMOS, Grade, experience, etc. 2) A code that represents a specific set of MOS's.
OSGM	Officer Staffing Goal Model - a computer program that takes an Inventory, ASR and Dictionary as input and returns an allocation of the officers to the billets.
PMOS	Primary MOS - the specialty that an officer will devote most of his/her career to.
PRIORITY	the relative importance of a billet descriptor for use in filling a billet. (PRIORITY = 1-5)
SPL	Staffing Precedence Level - a number reflecting the relative importance a command holds for filling its billets. (SPL = 1-5)
STAFFING GOAL DATE	the reference date on which a model run is based. It is the date the current officer inventory is produced.
WINDOW	the number of months before the run date in which all personnel whose current tours end are considered free.

APPENDIX B
PROTOTYPE PROGRAM REVIEW

PREPROCESSING:

NUMMOS FORTRAN	NUMBERS THE PRIMARY AND ADDITIONAL MOS's IN THE B1 FILE FOR USE IN LATER PROGRAMS. PERFORMANCE - 0.40 SEC
INVSPL FORTRAN	BREAKS THE ORIGINAL INVENTORY INTO MOVERS AND NONMOVERS, ACCORDING TO THE STAFF DATE AND WINDOW SIZE. PERFORMANCE: 58.33 SEC
BSORT SAS	SORTS THE NONMOVER INV FILE BY MCC, PMOS, GD. PERFORMANCE - 18.58 SEC
EXPE3 FORTRAN	EXPANDS THE E3 CARDS INTO A SINGLE LINE FORMAT. PERFORMANCE - 0.63 SEC
E3SORT SAS	SORT THE EXPANDED E3 CARDS BY CAT, MOS, GD. PERFORMANCE - 0.62 SEC
DIVE3 FORTRAN	DIVIDES THE E3 CARDS INTO TRAINING AND NONCHARGEABLE AND CONVERTS THE GRADE AND MOS. PERFORMANCE - 0.87 SEC
PULLE3T FORTRAN	REMOVES ANY PERSONNEL AT ANY F3 MCC E3T FILE. PERFORMANCE - 26.24 SEC

INTERMEDIATE PROCESSING:

MKCRIT FORTRAN	ALLOWS THE USER TO DEFINE THE GROUPS HE WISHES TO USE FOR THAT PARTICULAR RUN. PERFORMANCE - 0.02 SEC
GRNINV FORTRAN	GROUPS THE NONMOVER INVENTORY BY THE GROUPS SPECIFIED BY MKCRIT FORTRAN. PERFORMANCE - 28.95 SEC
GRMINV FORTRAN	GROUPS THE MOVER INVENTORY BY THE GROUPS SPECIFIED BY MKCRIT FORTRAN. PERFORMANCE - 15.31 SEC
E1SORT SAS	SORTS THE E1 CARDS BY BOD. PERFORMANCE - 3.44 SEC

EXPE1 FORTRAN	EXPANDS THE FULL E1 FILE BY ENUMERATING FROM LOGD TO HIGD. IT ALSO TRANSFORMS EXP AND UR. PERFORMANCE - 7.51 SEC
GRE1 FORTRAN	GROUPS THE E1 FILE ACCORDING TO GROUP CRIT. PERFORMANCE - 6.17 SEC
EXPASR FORTRAN	EXPANDS THE ASR FROM THE ORIGINAL FORMAT TO A SINGLE LINE FORMAT AND NUMBERS THEM. PERFORMANCE - 9.17 SEC
EXPC1 SAS	EXPANDS THE C1 CARDS FROM THE ORIGINAL FORMAT TO A SINGLE LINE FORMAT. PERFORMANCE - 0.15 SEC
ASORT SAS	SORTS EXP ASR INTO MOS GD MCC FORMAT. PERFORMANCE - 8.10 SEC
CSORT SAS	SORTS EXPC1 FILE INTO MOS GD MCC FORMAT. PERFORMANCE - 0.32 SEC
ROLLC1 FORTRAN	ROLLS UP ANY DUPLICATE C1 LINES IN THE C1 FILE. PEROFRMANCE - 0.51 SEC
ADJASR FORTRAN	APPLIES THE ADJUSTMENTS GIVEN IN THE C1 CARDS TO THE ASR. PERFORMANCE - 8.62 SEC
NUMASR FORTRAN	NUMBER THE ASR AND GENERATE A POINTER ARRAY. PERFORMANCE - 9.46 SEC
E2SORT SAS	NUMBERS AND SORTS THE E2 CARDS BY MOS,GD,MCC,SPL, BOD,MAC. PERFORMANCE - 4.83 SEC
BMIS FORTRAN	IDENTIFIES AND NUMBERS EACH SEPARATE BMIS FROM THE SORTED E2 FILE. PERFORMANCE - 6.72 SEC
REND FORTRAN	BREAKS THE BMIS CARDS INTO SPL GROUPS. PERFORMANCE - 7.32 SEC
BMIX FORTRAN	MERGES THE ASR AND VALID E2 CARDS. PERFORMANCE - 24.83 SEC
MSORT SAS	SORTS THE TOTAL MIX RECORDS BY BOD. PERFORMANCE - 8.88 SEC
DEMAND FORTRAN	MERGES THE E1 CARDS AND THE MERGED TOTAL MIX CARDS INTO A DEMAND FILE. PERFORMANCE - 40.93 SEC

NMATCH FORTRAN	MATCHES THE NONMOVERS TO THE DEMANDS. PERFORMANCE - 60.00 SEC (est)
MMATCH FORTRAN	MATCHES THE MOVERS TO THE DEMANDS. PERFORMANCE - 75.00 SEC (est)
FINAL PHASE :	
SHUFFLE SAS	SORTS THE MATCH DATA BY PMOS AND BMIS FOR THE SOLVER. PERFORMANCE - 50.00 SEC (est)
SOLVE FORTRAN	GNET SOLVER THAT WILL SOLVE EACH SPL IN TURN. PERFORMANCE - 180.00 SEC (est)
REPGEN FORTRAN	USES THE SOLVER RESULTS AND OTHER FILES TO GENERATE THE VARIOUS REPORTS REQUIRED. PERFORMANCE - 90.00 SEC (est)

APPENDIX C
INITIAL PROCESSING PROGRAMS

NUMMOS FORTRAN

```
* THIS PROGRAM TAKES THE B1 CARDS AND MAKES THREE FILES
* THAT WILL BE USED TO CONVERT THE MOS'S INTO MOS
* NUMBERS.  IT ALSO CREATES A FILE FOR INDICES FOR THE
* OFFICER TYPES INTO THE MOS NUMBERS.
      PROGRAM NUMMOS
      CHARACTER*4 MOS,DESC,OT(1:8,1:20)
      INTEGER MOSNUM,I,OTP(8)
      INTEGER OT1,OT2,OT3,OT4,OT5,OT6,OT7,OT8
      DATA OT1,OT2,OT3,OT4 /0,0,0,0 /
      DATA OT5,OT6,OT7,OT8 /0,0,0,0 /
      OPEN (UNIT = 1, FILE = 'FULL B1 A', STATUS = 'OLD')
      OPEN (UNIT = 2, FILE = 'PMOS B1 A', STATUS = 'NEW')
      OPEN (UNIT = 3, FILE = 'ADMOS B1 A', STATUS = 'NEW')
      OPEN (UNIT = 4, FILE = 'BMOS B1 A', STATUS = 'NEW')
      OPEN (UNIT = 7, FILE = 'OT PTR A', STATUS = 'NEW')
      MOSNUM = 0
* READ IN THE MOS AND DESCRIPTOR FROM THE B1 FILE AND
* CREATE THE PMOS, ADMOS AND BMOS FILES.  NOTE THAT
* THE BMOS FILE HAS ALL OF THE MOS'S, THE ADMOS FILE
* HAS ALL THE MOS'S EXCEPT 'BL' AND THE PMOS FILE HAS
* ALL EXCEPT THE 'BL' AND 'AD' MOS'S.
   10 READ(1,4,END = 100) MOS,DESC
      MOSNUM = MOSNUM + 1
      WRITE(4,5) MOS,MOSNUM
      IF (DESC .NE. 'BL  ') THEN
         WRITE(3,5) MOS,MOSNUM
         IF (DESC .NE. 'AD  ') THEN
            WRITE(2,5) MOS,MOSNUM
         END IF
      END IF
      GOTO 10
* RESET THE B1 FILE
  100 REWIND(1)
* READ IN THE MOS'S AND IGNORE THE 'BL', 'AD',
* AND 'NOSG' MOS'S.
* CREATE AN ARRAY OF MOS NUMBERS FOR THE MOS'S IN
* EACH OFFICER TYPE
  110 READ(1,4,END = 150) MOS,DESC
      IF (DESC .EQ. 'BL  ') THEN
      ELSE IF (DESC .EQ. 'AD  ') THEN
      ELSE IF (DESC .EQ. 'NOSG') THEN
      ELSE IF (DESC .EQ. 'NAFW') THEN
         OT1 = OT1 + 1
         OT(1,OT1) = MOS
      ELSE IF (DESC .EQ. 'NAHE') THEN
         OT2 = OT2 + 1
         OT(2,OT2) = MOS
      ELSE IF (DESC .EQ. 'NANF') THEN
         OT3 = OT3 + 1
         OT(3,OT3) = MOS
      ELSE IF (DESC .EQ. 'GDCB') THEN
         OT4 = OT4 + 1
         OT(4,OT4) = MOS
      ELSE IF (DESC .EQ. 'GDCS') THEN
         OT5 = OT5 + 1
         OT(5,OT5) = MOS
      ELSE IF (DESC .EQ. 'GDSS') THEN
         OT6 = OT6 + 1
         OT(6,OT6) = MOS
```

```
      ELSE IF (DESC .EQ. 'AGCS') THEN
         OT7 = OT7 + 1
         OT(7,OT7) = MOS
      ELSE IF (DESC .EQ. 'AGSS') THEN
         OT8 = OT8 + 1
         OT(8,OT8) = MOS
      END IF
      GOTO 110
  150 OTP(1) = OT1
      OTP(2) = OT2
      OTP(3) = OT3
      OTP(4) = OT4
      OTP(5) = OT5
      OTP(6) = OT6
      OTP(7) = OT7
      OTP(8) = OT8
* WRITE THE ARRAY TO THE OT PTR FILE
      DO 160 I = 1,8
         WRITE(7,6) OTP(I),OT(I,1),OT(I,2),OT(I,3),
     C               OT(I,4),OT(I,5),OT(I,6),OT(I,7),
     C               OT(I,8),OT(I,9),OT(I,10),OT(I,11),
     C               OT(I,12),OT(I,13),OT(I,14),OT(I,15)
  160 CONTINUE
    4 FORMAT(A4,1X,A4)
    5 FORMAT(A4,1X,I3)
    6 FORMAT(I4,15(1X,A4))
  200 STOP
      END
```

```
* THIS PROGRAM GETS THE STAFFING GOAL DATE AND WINDOW SIZE
* FROM THE USER AND THEN SPLITS THE INVENTORY INTO MOVERS
* AND NONMOVERS
* THE FUNCTION ADDS A DATE IN YYMM FORM TO A NUMBER
* OF MONTHS AND RETURNS THE RESULTS IN YYMM FORM
      INTEGER FUNCTION CDATE(NDATE,NCF)
                              INTEGER NDATE,NCF

      INTEGER T
      T = MOD(NDATE,100) + NCF
      CDATE = (INT(INT(NDATE/100)+ (T/12)) * 100)
     C         + MOD(T,12)
      END
* THE FUNCTION IS DONE

      PROGRAM INVSPL
      CHARACTER*1 SEX,JUNK
      CHARACTER*2 INGD,INCBGD,INFBGD
      CHARACTER*3 MCC,FMCC,AMCC
      INTEGER UR,EXP,DCTB,TCF,FTCF,ITD,FEDA
      INTEGER CBGD,FBGD,JMOS
      INTEGER AEDA,STDATE,WINDOW,I,GD,DE,AMT,LNUM
      INTEGER PMOS,AMOS1,AMOS2,CMOS,FMOS,INVMOS,MOSNUM
      INTEGER CDATE, INVDT,PM(0:9999), PA(0:9999), INVNUM
      DATA PM, PA / 10000*0, 10000*0 /
      OPEN (UNIT = 8, FILE = 'TEST INV A', STATUS = 'OLD')
      OPEN (UNIT = 9, FILE = 'FULL G1 A', STATUS = 'OLD')
      OPEN (UNIT=10, FILE='MOVER INV A',STATUS='NEW')
      OPEN (UNIT=11, FILE='NONMOVER INV A',STATUS='NEW')
      OPEN (UNIT = 12, FILE = 'PMOS B1 A', STATUS = 'OLD')
      OPEN (UNIT = 13, FILE = 'ADMOS B1 A', STATUS = 'OLD')
* GET STAFFING GOAL DATE AND WINDOW FROM THE USER
      WRITE(6,*) 'ENTER THE INVENTORY DATE: '
      READ(5,*) INVDT
      WRITE(6,*) 'ENTER THE WINDOW SIZE IN MONTHS(<= 12): '
      READ(5,*) WINDOW
      STDATE = CDATE(INVDT,WINDOW)
      INVNUM = 0
      JMOS = 0
* READ IN THE PMOS ARRAY DATA
  20  READ(12,17,END = 30) INMOS,MOSNUM
      PM(INMOS) = MOSNUM
      GOTO 20
* READ IN THE ADMOS ARRAY DATA
   30 READ(13,17,END = 5) INMOS,MOSNUM
      PA(INMOS) = MOSNUM
      GOTO 30
   17 FORMAT(I4,1X,I3)
* READ IN INVENTORY DATA, TEST FOR MOVER OR NONMOVER
* AND WRITE TO THE APPROPRIATE FILE.
    5 READ(8,2,END = 200) PMOS,INGD,AMOS1,AMOS2,UR,
     C    MCC,EXP,SEX,DCTB,TCF,ITD,FMCC,FEDA,AMCC,AEDA,
     C    INCBGD,CMOS,INFBGD,FMOS,FTCF
    2 FORMAT(I4,1X,A2,1X,I4,1X,I4,1X,I1,1X,A3,1X,I1,1X,A1,
     C       1X,I4,1X,I2,1X,I4,1X,A3,1X,I4,1X,A3,1X,I4,1X,
     C       A2,1X,I4,1X,A2,1X,I4,1X,I4)
* CHECK PMOS TO ENSURE THAT IT IS VALID
      IF (PM(PMOS) .EQ. 0) GOTO 5
* CONVERT THE GRADE TO A NUMBER FORMAT
      IF (INGD .EQ. '02') THEN
         GD = 2
      ELSE IF (INGD .EQ. '03') THEN
         GD = 3
      ELSE IF (INGD .EQ. '04') THEN
         GD = 4
      ELSE IF (INGD .EQ. '05') THEN
         GD = 5
      ELSE IF (INGD .EQ. 'WO') THEN
```

```
         GD = 1
      ELSE IF (INGD .EQ. '06') THEN
         GD = 6
      END IF
* CONVERT THE GD AT THE CURRENT MCC
      IF (INCBGD .EQ. '  ') GOTO 7
      IF (INCBGD .EQ. '02') THEN
         CBGD = 2
      ELSE IF (INCBGD .EQ. '03') THEN
         CBGD = 3
      ELSE IF (INCBGD .EQ. '04') THEN
         CBGD = 4
      ELSE IF (INCBGD .EQ. '05') THEN
         CBGD = 5
      ELSE IF (INCBGD .EQ. 'WO') THEN
         CBGD = 1
      ELSE IF (INCBGD .EQ. '06') THEN
         CBGD = 6
      END IF
* CONVERT THE GD AT THE FUTURE MCC
    7 IF (INFBGD .EQ. '  ') GOTO 8
      IF (INFBGD .EQ. '02') THEN
         FBGD = 2
      ELSE IF (INFBGD .EQ. '03') THEN
         FBGD = 3
      ELSE IF (INFBGD .EQ. '04') THEN
         FBGD = 4
      ELSE IF (INFBGD .EQ. '05') THEN
         FBGD = 5
      ELSE IF (INFBGD .EQ. 'WO') THEN
         FBGD = 1
      ELSE IF (INFBGD .EQ. '06') THEN
         FBGD = 6
      END IF
    8 INVNUM = INVNUM + 1
      IF (AMCC .NE. '   ' .AND. AEDA .LT. STDATE) THEN
         MCC = AMCC
         WRITE(11,3) PM(PMOS),GD,PA(AMOS1),PA(AMOS2),UR,
     C               MCC,EXP,SEX,JMOS,INVNUM
         GOTO 5
      ELSEIF (FMCC .NE. '   ' .AND. FEDA .LT. STDATE) THEN
            IF (CDATE(FEDA,FTCF) .LT. STDATE) THEN
               WRITE(10,3) PM(PMOS),GD,PA(AMOS1),PA(AMOS2),
     C                     UR,MCC,EXP,SEX,JMOS,INVNUM
            ELSE
               MCC = FMCC
               IF (INFBGD .NE. '  ' .AND.
     C             PM(FMOS) .NE. 0) THEN
                 PMOS = FMOS
                 GD = FBGD
               END IF
               WRITE(11,3) PM(PMOS),GD,PA(AMOS1),PA(AMOS2),
     C                     UR,MCC,EXP,SEX,JMOS,INVNUM
            ENDIF
            GOTO 5
      ELSEIF (MCC .NE. '   ' .AND. ITD .GE. STDATE) THEN
               IF (INCBGD .NE. '  ' .AND.
     C             PM(CMOS) .NE. 0) THEN
                 PMOS = CMOS
                 GD = CBGD
               END IF
            WRITE(11,3) PM(PMOS),GD,PA(AMOS1),PA(AMOS2),UR,
     C                  MCC,EXP,SEX,JMOS,INVNUM
            GOTO 5
      ELSEIF (MCC .NE. '   ' .AND.
     C        CDATE(DCTB,TCF) .GE. STDATE) THEN
               IF (INCBGD .NE. '  ' .AND.
     C             PM(CMOS) .NE. 0) THEN
                 PMOS = CMOS
                 GD = CBGD
```

```
            END IF
            WRITE(11,3) PM(PMOS),GD,PA(AMOS1),PA(AMOS2),
     C                  UR,MCC,EXP,SEX,JMOS,INVNUM
            GOTO 5
         ELSE
            WRITE(10,3) PM(PMOS),GD,PA(AMOS1),PA(AMOS2),
     C                  UR,MCC,EXP,SEX,JMOS,INVNUM
            GOTO 5
         END IF
         GOTO 5
* THIS SECTION HANDLES THE G1 CARDS
  200 AMOS1 = 0
         AMOS2 = 0
         MCC = 'ZZZ'
         LNUM = 0
   60 READ(9,4,END = 250) AMT,GD,DE,PMOS,SEX
    4 FORMAT(1X,I4,1X,I1,1X,I1,1X,I4,15X,A1)
         IF (PM(PMOS) .EQ. 0) GOTO 60
         IF (DE .EQ. 1) THEN
            UR = 2
            EXP = 0
         ELSE IF (DE .EQ. 2) THEN
            UR = 2
            EXP = 1
         ELSE IF (DE .EQ. 3) THEN
            UR = 1
            EXP = 0
         ELSE IF (DE .EQ. 4) THEN
            UR = 1
            EXP = 1
         END IF
         DO 40 I = 1,AMT
           INVNUM = INVNUM + 1
           WRITE(10,3) PM(PMOS),GD,PA(AMOS1),PA(AMOS2),UR,
     C                 MCC,EXP,SEX,JMOS,INVNUM
    3    FORMAT(I4,1X,I1,1X,I4,1X,I4,1X,I1,1X,A3,1X,I1,1X,
     C           A1,1X,I4,1X,I5)
   40 CONTINUE
         GOTO 60
  250 STOP
         END
```

BSORT SAS

```
* THIS PROGRAM SORTS THE NONMOVER FILE BY;
* MCC, PMOS, GD;
CMS FILEDEF FONE DISK NONMOVER INV A;
CMS FILEDEF FTWO DISK NMSORT INV A(RECFM F
    LRECL 80 BLOCK 80;
DATA DONE;
  INFILE FONE;
  INPUT PMOS 1-4 GD 6 AMOS1 $ 8-11 AMOS2 $ 13-16 UR $ 18
        MCC $ 20-22 EXP $ 24 SEX $ 26 JMOS $ 28-31
        NUM $ 33-37;
PROC SORT;
  BY MCC PMOS GD;
DATA _NULL_;
  SET DONE;
  FILE FTWO;
  PUT PMOS 1-4 GD 6 AMOS1 8-11 AMOS2 13-16 UR 18 MCC 20-22
      EXP 24 SEX 26 JMOS 28-31 NUM 33-37;
```

EXPE3 FORTRAN

```
C234567
* THIS PROGRAM WILL TAKE FULL E3 AND CONVERT IT
* TO A SINGLE LINE FORMAT
      PROGRAM EXPE3
      CHARACTER*1 CAT
      CHARACTER*2 GD
      CHARACTER*4 MOS1,MOS2,MOS3,MOS4,MOS5,MOS6
      INTEGER AMT1,AMT2,AMT3,AMT4,AMT5,AMT6
      OPEN (UNIT = 1, FILE = 'FULL E3 A', STATUS = 'OLD')
      OPEN (UNIT = 2, FILE = 'EXPE3 FILE A', STATUS = 'NEW')
* READ IN A LINE FROM FULL E3
   10 READ(1,3,END = 200) CAT,GD,MOS1,AMT1,MOS2,AMT2,MOS3,
     C                    AMT3,MOS4,AMT4,MOS5,AMT5,MOS6,AMT6
    3 FORMAT(A1,1X,A2,1X,A4,1X,I3,A4,1X,I3,A4,1X,I3,A4,
     C       1X,I3,A4,1X,I3,A4,1X,I3)
      WRITE(2,4) CAT,GD,MOS1,AMT1
      IF (MOS2 .EQ. '    ') GOTO 10
      WRITE(2,4) CAT,GD,MOS2,AMT2
      IF (MOS3 .EQ. '    ') GOTO 10
      WRITE(2,4) CAT,GD,MOS3,AMT3
      IF (MOS4 .EQ. '    ') GOTO 10
      WRITE(2,4) CAT,GD,MOS4,AMT4
      IF (MOS5 .EQ. '    ') GOTO 10
      WRITE(2,4) CAT,GD,MOS5,AMT5
      IF (MOS6 .EQ. '    ') GOTO 10
      WRITE(2,4) CAT,GD,MOS6,AMT6
      GOTO 10
    4 FORMAT(A1,1X,A2,1X,A4,1X,I3)
  200 STOP
      END
```

E3SORT SAS

```
* THIS PROGRAM SORTS THE EXPANDED E3 CARDS BY;
* CAT, MOS, GD;
CMS FILEDEF FONE DISK EXPE3 FILE A;
CMS FILEDEF FTWO DISK EXP E3 A(RECFM F LRECL 80 BLOCK 80;
DATA DONE;
  INFILE FONE;
  INPUT CAT $ 1 GD $ 3-4 MOS $ 6-9 AMT $ 11-13;
PROC SORT;
  BY CAT MOS GD;
DATA _NULL_;
  SET DONE;
  FILE FTWO;
  PUT CAT 1 GD 3-4 MOS 6-9 AMT 11-13;
```

```
* THIS PROGRAM PUTS THE E3 CARDS IN SINGLE LINE FORMAT AND
* SPLITS THEM INTO TWO FILES - TRAINING AND NON-CHARGEABLE
      PROGRAM DIVE3
      CHARACTER*1 TCAT,CAT
      CHARACTER*2 INGD,TGD
      INTEGER AMT,GD,MOS,PM(1:9999),MOSNUM,INMOS,TMOS,TAMT
      DATA PM / 9999*0 /
      OPEN (UNIT = 1, FILE = 'EXP E3 A', STATUS = 'OLD')
      OPEN (UNIT = 2, FILE = 'E3T FILE A', STATUS = 'NEW')
      OPEN (UNIT = 3, FILE = 'E3N FILE A', STATUS = 'NEW')
      OPEN (UNIT = 4, FILE = 'PMOS B1 A', STATUS = 'OLD')
* READ IN THE MOS ARRAY DATA
   10 READ(4,9,END = 15) INMOS,MOSNUM
      PM(INMOS) = MOSNUM
      GOTO 10
* READ IN AND ROLL UP THE E3'S
   15 READ(1,7,END = 200) TCAT,TGD,TMOS,TAMT
   20 READ(1,7,END = 200) CAT,INGD,INMOS,AMT
      IF (CAT .EQ. TCAT .AND. INGD .EQ. TGD .AND.
     C    INMOS .EQ. TMOS) THEN
          TAMT = TAMT + AMT
          GOTO 20
      ELSE
          IF (TGD .EQ. '03') THEN
             GD = 3
          ELSE IF (TGD .EQ. '02') THEN
             GD = 2
          ELSE IF (TGD .EQ. '04') THEN
             GD = 4
          ELSE IF (TGD .EQ. 'WO') THEN
             GD = 1
          ELSE IF (TGD .EQ. '05') THEN
             GD = 5
          ELSE IF (TGD .EQ. '06') THEN
             GD = 6
          END IF
          IF (TCAT .EQ. 'T') THEN
             WRITE (2,8) TCAT,GD,PM(TMOS),TAMT
          ELSE
             WRITE (3,8) TCAT,GD,PM(TMOS),TAMT
          END IF
      TCAT = CAT
      TGD = INGD
      TMOS = INMOS
      TAMT = AMT
      GOTO 20
      END IF
    7 FORMAT(A1,1X,A2,1X,I4,1X,I3)
    8 FORMAT(A1,1X,I1,1X,I4,1X,I3)
    9 FORMAT(I4,1X,I3)
  200 STOP
      END
```

```
* THIS PROGRAM PULLS THE E3 TRAINING PERSONNEL OUT OF
* THE NONMOVER INVENTORY.  THE MCC'S FOR THE E3T ARE IN
* THE F3 FILE.  THE PROGRAM PRODUCES A REDUCED NONMOVER
* FILE, E3 PERSONNEL FILE, SPL0 EXCESS FILE AND THE
* UNFILLED E3 BILLET FILE.  IT IS ASSUMED THAT THE F3
* FILE IS IN  SINGLE LINE FORMAT AND IS SORTED.
      PROGRAM PULLE3
      CHARACTER*1 SEX,UR,EXP
      CHARACTER*3 INVMCC,F3MCC,PRIMCC
      INTEGER E3AMT(0:229,1:6),TAMT,INVNUM,JMOS
      INTEGER INVGD,TGD,PMOS,AMOS1,AMOS2,TMOS,I,J
      DATA E3AMT / 1380*-1/
      OPEN (UNIT = 1, FILE = 'E3T FILE A', STATUS = 'OLD')
      OPEN (UNIT = 2, FILE = 'NMSORT INV A', STATUS = 'OLD')
      OPEN (UNIT = 3, FILE = 'FULL F3 A', STATUS = 'OLD')
      OPEN (UNIT = 7, FILE = 'REDUCED NONMOVER A', STATUS = 'NEW')
      OPEN (UNIT = 8, FILE = 'E3 PERS A', STATUS = 'NEW')
      OPEN (UNIT = 9, FILE = 'SPL0 EXCESS A', STATUS = 'NEW')
      OPEN (UNIT = 10, FILE = 'UNFILLED E3BIL A', STATUS = 'NEW')
      OPEN (UNIT = 11, FILE = 'USED F3 A', STATUS = 'NEW')
* READ THE E3T DATA INTO AN ARRAY
   10 READ(1,2,END = 20) TGD,TMOS,TAMT
      E3AMT(TMOS,TGD) = TAMT
      GOTO 10
   20 PRIMCC = '000'
      READ(3,3,END = 70) F3MCC
   30 READ(2,4,END = 200) PMOS,INVGD,AMOS1,AMOS2,UR,
     C                    INVMCC, EXP,SEX,JMOS,INVNUM
   40 IF (F3MCC .EQ. INVMCC) THEN
         IF (E3AMT(PMOS,INVGD) .GT. 0) THEN
            WRITE(8,4) PMOS,INVGD,AMOS1,AMOS2,UR,INVMCC,
     C                 EXP,SEX,JMOS,INVNUM
            IF (F3MCC .NE. PRIMCC) THEN
               WRITE(11,3) F3MCC
               PRIMCC = F3MCC
            END IF
            E3AMT(PMOS,INVGD) = E3AMT(PMOS,INVGD) -1
            GOTO 30
         ELSE IF (E3AMT(PMOS,INVGD) .EQ. 0) THEN
            WRITE(9,4) PMOS,INVGD,AMOS1,AMOS2,UR,INVMCC,
     C                 EXP,SEX,JMOS,INVNUM
            GOTO 30
         ELSE
            WRITE(7,4) PMOS,INVGD,AMOS1,AMOS2,UR,INVMCC,
     C                 EXP,SEX,JMOS,INVNUM
            GOTO 30
         END IF
       ELSE IF (F3MCC .GT. INVMCC) THEN
         WRITE(7,4) PMOS,INVGD,AMOS1,AMOS2,UR,INVMCC,
     C              EXP,SEX,JMOS,INVNUM
         GOTO 30
       ELSE
         READ(3,3,END = 70) F3MCC
         GOTO 40
       END IF
    70 READ(2,4,END = 200) PMOS,INVGD,AMOS1,AMOS2,UR,INVMCC,
     C                     EXP,SEX,JMOS,INVNUM
       WRITE(7,4) PMOS,INVGD,AMOS1,AMOS2,UR,INVMCC,EXP,SEX,
     C            JMOS,INVNUM
       GOTO 70
   200 DO 220 I = 1,229
         DO 210 J = 1,6
            IF (E3AMT(I,J) .GT. 0) THEN
               WRITE(10,2) J, I, E3AMT(I,J)
            END IF
   210    CONTINUE
   220 CONTINUE
     2 FORMAT(2X,I1,1X,I4,1X,I3)
```

```
3 FORMAT(A3)
4 FORMAT(I4,1X,I1,1X,I4,1X,I4,1X,A1,1X,A3,1X,A1,
 C       1X,A1,I4,1X,I5)
  STOP
  END
```

APPENDIX D
INTERMEDIATE PROCESSING PROGRAMS

MKCRIT FORTRAN

```
* THIS PROGRAM TAKES USER INPUT AND CREATES A FILE WITH
* CRITERIA THAT WILL BE USED TO GROUP THE INVENTORY IN A
* WAY THAT SHOULD REDUCE SEARCH TIME WHEN MAKING MATCHES.
*
      PROGRAM MKCRIT
      CHARACTER*1 CHVAL,URANS,EXPANS,SEXANS,MORE
      CHARACTER*6 CRNAME
      INTEGER CRNUM,CHNUM,I,K,PRENUM
      OPEN (UNIT=1, FILE = 'GROUP CRIT A', STATUS = 'NEW')
* ASK USER HOW MANY DIFFERENT CRITERIA THEY WISH
* TO USE IN GROUPING
      WRITE(6,*) 'THIS PROGRAM TAKES YOUR INPUT AND'
      WRITE(6,*) 'PRODUCES GROUPING CRITERIA USED IN THE'
      WRITE(6,*) 'GROUPING OF THE INVENTORY INTO MUTUALLY'
      WRITE(6,*) 'EXCLUSIVE GROUPS.  THIS SHOULD REDUCE '
      WRITE(6,*) 'THE AMOUNT OF TIME NEEDED TO MATCH'
      WRITE(6,*) 'PEOPLE TO BILLETS.  ENTER THE'
      WRITE(6,*) 'NUMBER OF CRITERIA YOU WISH TO USE'
      WRITE(6,*) 'FOR THIS PARTICULAR PREPROCESSING RUN.'
      WRITE(6,*) 'REMEMBER THAT THAT THE TOTAL NUMBER OF'
      WRITE(6,*) 'GROUPS CREATED WILL BE THE'
      WRITE(6,*) 'CROSS PRODUCT OF THE NUMBER OF CHOICES'
      WRITE(6,*) 'FOR EACH CRITERIA.  TOO '
      WRITE(6,*) 'MANY GROUPS (>60) WILL MAKE THE ENTIRE'
      WRITE(6,*) 'MATCHING PROCESS VERY, VERY SLOW.'
      WRITE(6,*) '                                 '
      WRITE(6,*) 'PLEASE MAKE SURE THAT ALL CHARACTER'
      WRITE(6,*) 'ANSWERS ARE UPPER CASE - THANK YOU'
      PRENUM = 1
      WRITE (6,*) 'USE RESTR (UR) AS A CRITERIA (Y/N) ?'
      READ(5,5) URANS
      WRITE (1,5) URANS
      WRITE(6,*) 'EXPERIENCE (EXP) AS A CRITERIA (Y/N) ?'
      READ(5,5) EXPANS
      WRITE (1,5) EXPANS
      WRITE(6,*) 'USE SEX AS A CRITERIA (Y/N) ?'
      READ(5,5) SEXANS
      WRITE (1,5) SEXANS
   15 WRITE(6,*) 'USE ANY ADDITIONAL CRITERIA (Y/N) ?'
      READ(5,5) MORE
      WRITE (1,5) MORE
      IF (MORE .EQ. 'Y') THEN
        WRITE (6,*) 'ENTER THE NAME OF THE CRITERIA '
        READ(5,4) CRNAME
        WRITE (1,4) CRNAME
    4   FORMAT(A6)
        WRITE (6,*) 'ENTER THE NUMBER OF CHOICES
     C                FOR THE CRITERIA'
        READ(5,2) CHNUM
        PRENUM = PRENUM * CHNUM
        WRITE (1,2) CHNUM
    2   FORMAT(I1)
        DO 20 K = 1,CHNUM
          WRITE(6,*) 'ENTER THE VALUE FOR CHOICE #',K
          READ(5,3) CHVAL
          WRITE(1,3) CHVAL
    3     FORMAT(A1)
   20   CONTINUE
      GOTO 15
```

```
    END IF
  5 FORMAT (A1)
200 STOP
    END
```

```
* THIS PROGRAM CLASSIFIES EACH ENTRY IN THE INVENTORY
* BY ITS GROUPING CATEGORY AND ITS MOS CATEGORY.
* THE DATA FOR THE GROUPING COMES FROM
* THE 'GROUP CRIT' FILE.  THE MOS CATEGORIES ARE:
* PMOS ONLY, PMOS AND ONE AMOS, PMOS AND TWO AMOS.
      PROGRAM GRINV
      CHARACTER*1 EXP,SEX,UR,E(2),S(2),U(2),EXPANS
      CHARACTER*1 SEXANS,URANS,CHVAL,V1(64,4)
      CHARACTER*1 MORE,SMORE,B1(4),B2(4),V(5)
      CHARACTER*1 CGROUP,PGROUP,SV(128,5),BLANK1,BLANK2
      CHARACTER*3 MCC
      CHARACTER*5 TANS
      CHARACTER*6 CRNAM1, CRNAM2
      INTEGER GD,PMOS,AMOS1,AMOS2,INVNUM
      INTEGER I, J, K, L, M, N, CHNUM1, CHNUM2
      INTEGER NC(5),C1,C2,C3,C4,C5
      OPEN (UNIT=1, FILE = 'GROUP CRIT A', STATUS = 'OLD')
      OPEN (UNIT=2, FILE = 'NMSORT INV A', STATUS = 'OLD')
      OPEN (UNIT=4, FILE = 'RNMG FILE A', STATUS = 'NEW')
      OPEN (UNIT=7, FILE='RNMG PAMOS1 A', STATUS = 'NEW')
      OPEN (UNIT=8, FILE='RNMG PAMOS2 A', STATUS = 'NEW')
* SET THE STANDARD VALUES
      SV(2,1) = '0'
      SV(2,2) = '1'
      SV(1,1) = '1'
      SV(1,2) = '2'
      SV(3,1) = 'M'
      SV(3,2) = 'F'
      C1 = 0
      C2 = 0
      C3 = 0
      C4 = 0
      C5 = 0
* READ IN DATA FROM GROUP CRIT FILE
      READ(1,2) EXPANS
      READ(1,2) URANS
      READ(1,2) SEXANS
      TANS = 'NNNNN'
      TANS(1:1) = EXPANS
      TANS(2:2) = URANS
      TANS(3:3) = SEXANS
      N = 0
      IF (EXPANS .EQ. 'Y') N = N + 1
      IF (URANS .EQ. 'Y') N = N + 1
      IF (SEXANS .EQ. 'Y') N = N + 1
      READ(1,2) MORE
      TANS(4:4) = MORE
      IF (MORE .EQ. 'Y') THEN N = N + 1
* CHECK FOR BLANK1 DATA
      IF (MORE .EQ. 'Y') THEN
         READ(1,3) CRNAM1
         READ(1,5) CHNUM1
         DO 15 J = 1,CHNUM1
            READ(1,4) CHVAL
            B1(J) = CHVAL
 15      CONTINUE
         READ(1,2) SMORE
         TANS(5:5) = SMORE
         IF (SMORE .EQ. 'Y') THEN N = N + 1
  2      FORMAT(A1)
* CHECK FOR BLANK2 DATA
         IF (SMORE .EQ. 'Y') THEN
            READ(1,3) CRNAM2
  3         FORMAT(A6)
            READ(1,5) CHNUM2
            DO 20 J = 1,CHNUM2
               READ(1,4) CHVAL
  4            FORMAT(A1)
```

```
    5          FORMAT(I1)
               B2(J) = CHVAL
   20       CONTINUE
         END IF
      END IF
      WRITE(6,*) 'THE VALUE OF N = ',N
* MAKE THE BASIC VALUE ARRAY AND SET GROUPS
      IF (TANS(4:5) .NE. 'NN') GOTO 200
         IF (EXPANS .EQ. 'Y') THEN
            C1 = 1
            IF (URANS .EQ. 'Y') THEN
               C2 = 2
               IF (SEXANS .EQ. 'Y') C3 = 3
            ELSE
               IF (SEXANS .EQ. 'Y') C2 = 3
            ENDIF
         ELSE
            IF (URANS .EQ. 'Y') THEN
               C1 = 2
               IF (SEXANS .EQ. 'Y') C2 = 3
            ELSE
               IF (SEXANS .EQ. 'Y') C1 = 3
            ENDIF
          END IF
          WRITE (6,*) 'C1 = ',C1
          WRITE (6,*) 'C2 = ',C2
          WRITE (6,*) 'C3 = ',C3
          WRITE (6,*) 'C4 = ',C4
          WRITE (6,*) 'C5 = ',C5
* SET GROUPS FOR BASIC THREE
          IF (N .EQ. 1) THEN
   70        READ(2,11,END = 200) PMOS,GD,AMOS1,AMOS2,
     C                        V(1),MCC,V(2)V(3),JMOS,INVNUM
             IF (V(C1) .EQ. SV(C1,1)) THEN
                CGROUP = '1'
             ELSE
                CGROUP = '2'
             END IF
             IF ((AMOS1 .EQ. 0) .AND. (AMOS2 .EQ. 0)) THEN
             WRITE(4,12) PMOS,GD,AMOS1,AMOS2,V(1),MCC,V(2),
     C                   V(3),JMOS,INVNUM,CGROUP
             END IF
             IF (((AMOS1 .NE. 0) .AND. (AMOS2 .EQ. 0)) .OR.
     C          ((AMOS1 .EQ. 0) .AND. (AMOS2 .NE. 0))) THEN
             WRITE(7,12) PMOS,GD,AMOS1,AMOS2,V(1),MCC,V(2),
     C                   V(3),JMOS,INVNUM,CGROUP
             END IF
             IF ((AMOS1 .NE. 0) .AND. (AMOS2 .NE. 0)) THEN
             WRITE(8,12) PMOS,GD,AMOS1,AMOS2,V(1),MCC,
     C                 V(2),V(3),JMOS,INVNUM,CGROUP
             END IF
             GOTO 70
          ELSE IF (N .EQ. 2) THEN
   80        READ(2,11,END = 200) PMOS,GD,AMOS1,AMOS2,
     C                       V(1),MCC,V(2),V(3),JMOS,INVNUM
             IF (V(C1) .EQ. SV(C1,1)) THEN
                IF (V(C2) .EQ. SV(C2,1)) THEN
                   CGROUP = '1'
                ELSE
                   CGROUP = '2'
                END IF
             ELSE
                IF (V(C2) .EQ. SV(C2,1)) THEN
                   CGROUP = '3'
                ELSE
                   CGROUP = '4'
                END IF
             END IF
             IF ((AMOS1 .EQ. 0) .AND. (AMOS2 .EQ. 0)) THEN
             WRITE(4,12) PMOS,GD,AMOS1,AMOS2,V(1),MCC,
```

```
     C                    V(2),V(3),JMOS,INVNUM,CGROUP
           END IF
           IF (((AMOS1 .NE. 0) .AND. (AMOS2 .EQ. 0)) .OR.
     C        ((AMOS1 .EQ. 0) .AND. (AMOS2 .NE. 0))) THEN
           WRITE(7,12) PMOS,GD,AMOS1,AMOS2,V(1),MCC,V(2),
     C                 V(3),JMOS,INVNUM,CGROUP
           END IF
           IF ((AMOS1 .NE. 0) .AND. (AMOS2 .NE. 0)) THEN
           WRITE(8,12) PMOS,GD,AMOS1,AMOS2,V(1),MCC,V(2),
     C                 V(3),JMOS,INVNUM,CGROUP
           END IF
           GOTO 80
        ELSE
  90       READ(2,11,END = 200) PMOS,GD,AMOS1,AMOS2,
     C                 V(1),MCC,V(2),V(3),JMOS,INVNUM
  11       FORMAT(I4,1X,I1,1X,I4,1X,I4,1X,A1,1X,A3,1X,
     C           A1,1X,A1,1X,I4,1X,I5)
           IF (V(C1) .EQ. SV(C1,1)) THEN
              IF (V(C2) .EQ. SV(C2,1)) THEN
                 IF (V(C3) .EQ. SV(C3,1)) THEN
                    CGROUP = '1'
                 ELSE
                    CGROUP = '2'
                 END IF
              ELSE
                 IF (V(C3) .EQ. SV(C3,1)) THEN
                    CGROUP = '3'
                 ELSE
                    CGROUP = '4'
                 END IF
              END IF
           ELSE
              IF (V(C2) .EQ. SV(C2,1)) THEN
                 IF (V(C3) .EQ. SV(C3,1)) THEN
                    CGROUP = '5'
                 ELSE
                    CGROUP = '6'
                 END IF
              ELSE
                 IF (V(C3) .EQ. SV(C3,1)) THEN
                    CGROUP = '7'
                 ELSE
                    CGROUP = '8'
                 END IF
              END IF
           END IF
           IF ((AMOS1 .EQ. 0) .AND. (AMOS2 .EQ. 0)) THEN
           WRITE(4,12) PMOS,GD,AMOS1,AMOS2,V(1),MCC,V(2),
     C                 V(3),JMOS,INVNUM,CGROUP
           END IF
           IF (((AMOS1 .NE. 0) .AND. (AMOS2 .EQ. 0)) .OR.
     C        ((AMOS1 .EQ. 0) .AND. (AMOS2 .NE. 0))) THEN
           WRITE(7,12) PMOS,GD,AMOS1,AMOS2,V(1),MCC,V(2),
     C                 V(3),JMOS,INVNUM,CGROUP
           END IF
           IF ((AMOS1 .NE. 0) .AND. (AMOS2 .NE. 0)) THEN
           WRITE(8,12) PMOS,GD,AMOS1,AMOS2,V(1),MCC,V(2),
     C                 V(3),JMOS,INVNUM,CGROUP
           END IF
  12       FORMAT(I4,1X,I1,1X,I4,1X,I4,1X,A1,1X,A3,1X,
     C           A1,1X,A1,1X,I4,1X,I5,1X,A1)
           GOTO 90
        END IF
200 STOP
    END
```

```
* THIS PROGRAM CLASSIFIES EACH ENTRY IN THE INVENTORY
* BY ITS GROUPING AND MOS CATEGORY.
* THE DATA FOR THE GROUPING COMES FROM
* THE 'GROUP CRIT' FILE.  THE MOS CATEGORIES ARE:
* PMOS ONLY, PMOS AND ONE AMOS, PMOS AND TWO AMOS.

      PROGRAM GRINV
      CHARACTER*1 EXP,SEX,UR,E(2),S(2),U(2),EXPANS
      CHARACTER*1 MORE,SMORE,B1(4),B2(4),V(5)
      CHARACTER*1 SEXANS,URANS,CHVAL,V1(64,4)
      CHARACTER*1 CGROUP,PGROUP,SV(128,5),BLANK1,BLANK2
      CHARACTER*3 MCC
      CHARACTER*5 TANS
      CHARACTER*6 CRNAM1, CRNAM2
      INTEGER GD,PMOS,AMOS1,AMOS2,INVNUM
      INTEGER I, J, K, L, M, N, CHNUM1, CHNUM2
      INTEGER NC(5),C1,C2,C3,C4,C5
      OPEN (UNIT = 1, FILE = 'GROUP CRIT A', STATUS = 'OLD')
      OPEN (UNIT = 2, FILE = 'MOVER INV A', STATUS = 'OLD')
      OPEN (UNIT = 4, FILE = 'MG FILE A', STATUS = 'NEW')
      OPEN (UNIT = 7, FILE = 'MG PAMOS1 A', STATUS = 'NEW')
      OPEN (UNIT = 8, FILE = 'MG PAMOS2 A', STATUS = 'NEW')
* SET THE STANDARD VALUES
      SV(2,1) = '0'
      SV(2,2) = '1'
      SV(1,1) = '1'
      SV(1,2) = '2'
      SV(3,1) = 'M'
      SV(3,2) = 'F'
      C1 = 0
      C2 = 0
      C3 = 0
      C4 = 0
      C5 = 0
* READ IN DATA FROM GROUP CRIT FILE
      READ(1,2) EXPANS
      READ(1,2) URANS
      READ(1,2) SEXANS
      TANS = 'NNNNN'
      TANS(1:1) = EXPANS
      TANS(2:2) = URANS
      TANS(3:3) = SEXANS
      N = 0
      IF (EXPANS .EQ. 'Y') N = N + 1
      IF (URANS .EQ. 'Y') N = N + 1
      IF (SEXANS .EQ. 'Y') N = N + 1
      READ(1,2) MORE
      TANS(4:4) = MORE
      IF (MORE .EQ. 'Y') THEN N = N + 1
* CHECK FOR BLANK1 DATA
      IF (MORE .EQ. 'Y') THEN
         READ(1,3) CRNAM1
         READ(1,5) CHNUM1
         DO 15 J = 1,CHNUM1
            READ(1,4) CHVAL
            B1(J) = CHVAL
   15    CONTINUE
         READ(1,2) SMORE
         TANS(5:5) = SMORE
         IF (SMORE .EQ. 'Y') THEN N = N + 1
    2    FORMAT(A1)
* CHECK FOR BLANK2 DATA
         IF (SMORE .EQ. 'Y') THEN
            READ(1,3) CRNAM2
    3       FORMAT(A6)
            READ(1,5) CHNUM2
            DO 20 J = 1,CHNUM2
               READ(1,4) CHVAL
```

```
     4          FORMAT(A1)
     5          FORMAT(I1)
                B2(J) = CHVAL
    20       CONTINUE
          END IF
       END IF
       WRITE(6,*) 'THE VALUE OF N = ',N
* MAKE THE BASIC VALUE ARRAY AND SET GROUPS
       IF (TANS(4:5) .NE. 'NN') GOTO 200
          IF (EXPANS .EQ. 'Y') THEN
             C1 = 1
             IF (URANS .EQ. 'Y') THEN
                C2 = 2
                IF (SEXANS .EQ. 'Y') C3 = 3
             ELSE
                IF (SEXANS .EQ. 'Y') C2 = 3
             ENDIF
          ELSE
             IF (URANS .EQ. 'Y') THEN
                C1 = 2
                IF (SEXANS .EQ. 'Y') C2 = 3
             ELSE
                IF (SEXANS .EQ. 'Y') C1 = 3
             ENDIF
           END IF
           WRITE (6,*) 'C1 = ',C1
           WRITE (6,*) 'C2 = ',C2
           WRITE (6,*) 'C3 = ',C3
           WRITE (6,*) 'C4 = ',C4
           WRITE (6,*) 'C5 = ',C5
* SET GROUPS FOR BASIC THREE
           IF (N .EQ. 1) THEN
    70        READ(2,11,END = 200) PMOS,GD,AMOS1,AMOS2,
     C                     V(1),MCC,V(2),V(3),JMOS,INVNUM
              IF (V(C1) .EQ. SV(C1,1)) THEN
                 CGROUP = '1'
              ELSE
                 CGROUP = '2'
              END IF
              IF ((AMOS1 .EQ. 0) .AND. (AMOS2 .EQ. 0)) THEN
              WRITE(4,12) PMOS,GD,AMOS1,AMOS2,V(1),MCC,V(2),
     C                     V(3),JMOS,INVNUM,CGROUP
              END IF
              IF (((AMOS1 .NE. 0) .AND. (AMOS2 .EQ. 0)) .OR.
     C            ((AMOS1 .EQ. 0) .AND. (AMOS2 .NE. 0))) THEN
              WRITE(7,12) PMOS,GD,AMOS1,AMOS2,V(1),MCC,V(2),
     C                     V(3),JMOS,INVNUM,CGROUP
              END IF
              IF ((AMOS1 .NE. 0) .AND. (AMOS2 .NE. 0)) THEN
              WRITE(8,12) PMOS,GD,AMOS1,AMOS2,V(1),MCC,V(2),
     C                     V(3),JMOS,INVNUM,CGROUP
              END IF
              GOTO 70
           ELSE IF (N .EQ. 2) THEN
    80        READ(2,11,END = 200) PMOS,GD,AMOS1,AMOS2,
     C                     V(1),MCC,V(2),V(3),JMOS,INVNUM
              IF (V(C1) .EQ. SV(C1,1)) THEN
                 IF (V(C2) .EQ. SV(C2,1)) THEN
                    CGROUP = '1'
                 ELSE
                    CGROUP = '2'
                 END IF
              ELSE
                 IF (V(C2) .EQ. SV(C2,1)) THEN
                    CGROUP = '3'
                 ELSE
                    CGROUP = '4'
                 END IF
              END IF
              IF ((AMOS1 .EQ. 0) .AND. (AMOS2 .EQ. 0)) THEN
```

```
            WRITE(4,12) PMOS,GD,AMOS1,AMOS2,V(1),MCC,V(2),
     C                  V(3),JMOS,INVNUM,CGROUP
            END IF
            IF (((AMOS1 .NE. 0) .AND. (AMOS2 .EQ. 0)) .OR.
     C          ((AMOS1 .EQ. 0) .AND. (AMOS2 .NE. 0))) THEN
            WRITE(7,12) PMOS,GD,AMOS1,AMOS2,V(1),MCC,V(2),
     C                  V(3),JMOS,INVNUM,CGROUP
            END IF
            IF ((AMOS1 .NE. 0) .AND. (AMOS2 .NE. 0)) THEN
            WRITE(8,12) PMOS,GD,AMOS1,AMOS2,V(1),MCC,V(2),
     C                  V(3),JMOS,INVNUM,CGROUP
            END IF
            GOTO 80
         ELSE
   90       READ(2,11,END = 200) PMOS,GD,AMOS1,AMOS2,
     C                  V(1),MCC,V(2),V(3),JMOS,INVNUM
   11       FORMAT(I4,1X,I1,1X,I4,1X,I4,1X,A1,1X,A3,1X,
     C               A1,1X,A1,1X,I4,1X,I5)
            IF (V(C1) .EQ. SV(C1,1)) THEN
               IF (V(C2) .EQ. SV(C2,1)) THEN
                  IF (V(C3) .EQ. SV(C3,1)) THEN
                     CGROUP = '1'
                  ELSE
                     CGROUP = '2'
                  END IF
               ELSE
                  IF (V(C3) .EQ. SV(C3,1)) THEN
                     CGROUP = '3'
                  ELSE
                     CGROUP = '4'
                  END IF
               END IF
            ELSE
               IF (V(C2) .EQ. SV(C2,1)) THEN
                  IF (V(C3) .EQ. SV(C3,1)) THEN
                     CGROUP = '5'
                  ELSE
                     CGROUP = '6'
                  END IF
               ELSE
                  IF (V(C3) .EQ. SV(C3,1)) THEN
                     CGROUP = '7'
                  ELSE
                     CGROUP = '8'
                  END IF
               END IF
            END IF
            IF ((AMOS1 .EQ. 0) .AND. (AMOS2 .EQ. 0)) THEN
            WRITE(4,12) PMOS,GD,AMOS1,AMOS2,V(1),MCC,V(2),
     C                  V(3),JMOS,INVNUM,CGROUP
            END IF
            IF (((AMOS1 .NE. 0) .AND. (AMOS2 .EQ. 0)) .OR.
     C          ((AMOS1 .EQ. 0) .AND. (AMOS2 .NE. 0))) THEN
            WRITE(7,12) PMOS,GD,AMOS1,AMOS2,V(1),MCC,V(2),
     C                  V(3),JMOS,INVNUM,CGROUP
            END IF
            IF ((AMOS1 .NE. 0) .AND. (AMOS2 .NE. 0)) THEN
            WRITE(8,12) PMOS,GD,AMOS1,AMOS2,V(1),MCC,V(2),
     C                  V(3),JMOS,INVNUM,CGROUP
            END IF
   12       FORMAT(I4,1X,I1,1X,I4,1X,I4,1X,A1,1X,A3,1X,
     C               A1,1X,A1,1X,I4,1X,I5,1X,A1)
            GOTO 90
         END IF
  200 STOP
      END
```

E1SORT SAS

```
* THIS PROGRAM SORTS THE E1 CARDS BY BOD;
CMS FILEDEF FONE DISK EXPANDED E1 A;
CMS FILEDEF FTWO DISK EXP E1 A(RECFM F LRECL 80 BLOCK 80;
DATA DONE;
   INFILE FONE;
   INPUT BOD $ 6-14 PRI $ 16 GD $ 18-19 EXP $ 21
         MOS $ 23-26 OT $ 28-35 AMOS1 $ 37-40 AMOS2 $ 42-45
         UR $ 47 SEX $ 49 MAC $ 51-58;
PROC SORT;
   BY BOD;
DATA _NULL_;
   SET DONE;
   FILE FTWO;
   PUT BOD 6-14 PRI 16 GD 18-19 EXP 21 MOS 23-26
       OT 28-35 AMOS1 37-40 AMOS2 42-45 UR 47
       SEX 49 MAC 51-58;
```

EXPE1 FORTRAN

```
* THIS PROGRAM EXPANDS THE E1 CARDS THAT HAVE MORE THAN
* ONE GRADE PER CARD.  IT ALSO TRANSFORMS THE EXP AND UR
* UR FIELDS TO BE THE SAME AS THE INVENTORY.
* HERE IS THE FUNCTION CNUM THAT RETURNS THE NUMBER
* VALUE OF A STRING
      INTEGER FUNCTION CNUM (CMOS)
                            CHARACTER*4 CMOS
      INTEGER T(1:4),I
      CHARACTER*1 C(1:4)
      C(1) = CMOS(1:1)
      C(2) = CMOS(2:2)
      C(3) = CMOS(3:3)
      C(4) = CMOS(4:4)
      DO 23 I = 1,4
         IF (C(I) .EQ. '*') THEN
            T(I) = 0
         ELSE IF (C(I) .EQ. '1') THEN
            T(I) = 1
         ELSE IF (C(I) .EQ. '2') THEN
            T(I) = 2
         ELSE IF (C(I) .EQ. '3') THEN
            T(I) = 3
         ELSE IF (C(I) .EQ. '4') THEN
            T(I) = 4
         ELSE IF (C(I) .EQ. '5') THEN
            T(I) = 5
         ELSE IF (C(I) .EQ. '6') THEN
            T(I) = 6
         ELSE IF (C(I) .EQ. '7') THEN
            T(I) = 7
         ELSE IF (C(I) .EQ. '8') THEN
            T(I) = 8
         ELSE IF (C(I) .EQ. '9') THEN
            T(I) = 9
         ELSE IF (C(I) .EQ. '0') THEN
            T(I) = 0
         END IF
   23 CONTINUE
      CNUM = T(1)*1000 + T(2)*100 + T(3)*10 + T(4)
      END
* THIS IS THE END OF CNUM
      PROGRAM EXPE1
      CHARACTER*1 HIEXP,LOEXP,SEX,UR,PRI,EXP
      CHARACTER*1 P1,P3,P4,A11,A13,A14,A21,A23,A24
      CHARACTER*2 HIGD,LOGD,GD,G(6)
      CHARACTER*4 INPMOS,INAM1,INAM2
      CHARACTER*8 OT,MAC
      CHARACTER*9 BOD
      INTEGER I,N,F,B,PM(0:9999),PA(0:9999),PMOS,AMOS1
      INTEGER CNUM,E1NUM,OGD,AMOS2
      DATA PM, PA / 10000*0, 10000*0 /
      OPEN (UNIT = 1, FILE = 'FULL E1 A', STATUS = 'OLD')
      OPEN (UNIT = 2, FILE = 'EXPANDED E1 A', STATUS = 'NEW')
      OPEN (UNIT = 3, FILE = 'PMOS B1 A', STATUS = 'OLD')
      OPEN (UNIT = 4, FILE = 'ADMOS B1 A', STATUS = 'OLD')
      E1NUM = 0
* READ IN THE MOS ARRAYS
   10 READ(3,7,END = 20) INMOS,MOSNUM
      PM(INMOS) = MOSNUM
      GOTO 10
   20 READ(4,7,END = 25) INMOS,MOSNUM
      PA(INMOS) = MOSNUM
      GOTO 20
C SET UP THE GRADE ARRAY
   25 G(1) = 'WO'
      G(2) = 'O2'
      G(3) = 'O3'
      G(4) = 'O4'
```

```
      G(5) = '05'
      G(6) = '06'
C READ IN ORIGINAL E1 DATA
   30 READ (1,3,END = 200) BOD,PRI,LOGD,LOEXP,HIGD,HIEXP,
     C                INPMOS,OT,INAM1,INAM2,UR,SEX,MAC
    3 FORMAT (5X,A9,1X,A1,1X,A2,1X,A1,1X,A2,1X,A1,1X,A4,1X,
     C        A8,1X,A4,1X,A4,1X,A1,1X,A1,1X,A8)
C TAKE OUT THE PRIORITY 4 AND 5'S
      IF (PRI .EQ. '4' .OR. PRI .EQ. '5') GOTO 30
C CONVERT THE EXPERIENCE TO A NUMBER
      IF (LOEXP .EQ. 'N') LOEXP = '0'
      IF (LOEXP .EQ. 'E') LOEXP = '1'
* CONVERT THE USE RESTRICTION TO A NUMBER
      IF (UR .EQ. 'U') UR = '1'
      IF (UR .EQ. 'L') UR = '2'
* CONVERT THE MOS TO AN INDEX
      P1 = INPMOS(1:1)
      P3 = INPMOS(3:3)
      P4 = INPMOS(4:4)
      A11 = INAM1(1:1)
      A13 = INAM1(3:3)
      A14 = INAM1(4:4)
      A21 = INAM2(1:1)
      A23 = INAM2(3:3)
      A24 = INAM2(4:4)
* PMOS = 240  - AMOS ONLY E1 CARD
* PMOS = 239  - OFFICER TYPE E1 CARD
* PMOS > 1000 - OCC FIELD OR SUBOCC FIELD PMOS E1 CARD
* AMOS > 1000 - OCC FIELD OR SUBOCC FIELD AMOS E1 CARD
      IF (P1 .NE. '*' .AND. P4 .NE. ' ') THEN
         PMOS = PM(CNUM(INPMOS))
      ELSE IF (P1 .EQ. '*') THEN
         PMOS = 240
      ELSE IF (P1 .EQ. ' ') THEN
         PMOS = 239
      ELSE IF (P3 .EQ. '*' .AND. P4 .EQ. '*') THEN
         PMOS = CNUM(INPMOS) + 1000
      ELSE IF (P4 .EQ. '*') THEN
         PMOS = CNUM(INPMOS) + 1000
      END IF
* CONVERT AMOS1
      IF (A14 .EQ. '*' .AND. A11 .EQ. '*') THEN
         AMOS1 = 0
      ELSE IF (A13 .EQ. '*' .AND. A14 .EQ. '*') THEN
         AMOS1 = CNUM(INAM1) + 1000
      ELSE IF (A14 .EQ. '*') THEN
         AMOS1 = CNUM(INAM1) + 1000
      ELSE IF (A11 .NE. '*' .AND. A14 .NE. '*') THEN
         AMOS1 = PA(CNUM(INAM1))
      END IF
* CONVERT AMOS2
      IF (A24 .EQ. '*' .AND. A21 .EQ. '*') THEN
         AMOS2 = 0
      ELSE IF (A23 .EQ. '*' .AND. A24 .EQ. '*') THEN
         AMOS2 = CNUM(INAM1) + 1000
      ELSE IF (A24 .EQ. '*') THEN
         AMOS2 = CNUM(INAM1) + 1000
      ELSE IF (A21 .NE. '*' .AND. A24 .NE. '*') THEN
         AMOS2 = PA(CNUM(INAM2))
      END IF
* EXPAND THE GRADE AND EXPERIENCE
      IF (HIGD .NE. '  ') THEN
         IF (HIEXP .EQ. 'N') HIEXP = '0'
         IF (HIEXP .EQ. 'E') HIEXP = '1'
         DO 40 I = 1,6
            IF (LOGD .EQ. G(I)) F = I
   40    CONTINUE
         DO 50 I = 1,6
            IF (HIGD .EQ. G(I)) B = I
   50    CONTINUE
```

```
      DO 60 I = F,B
         IF (I .NE. B) THEN
            EXP = LOEXP
         ELSE
            EXP = HIEXP
         END IF
         E1NUM = E1NUM + 1
         WRITE (2,4) E1NUM,BOD,PRI,I,EXP,PMOS,
     C                  OT,AMOS1,AMOS2,UR,SEX,MAC
60    CONTINUE
      GOTO 30
   END IF
   E1NUM = E1NUM + 1
   IF (LOGD .EQ. '03') THEN
      OGD = 3
   ELSE IF (LOGD .EQ. '02') THEN
      OGD = 2
   ELSE IF (LOGD .EQ. '04') THEN
      OGD = 4
   ELSE IF (LOGD .EQ. 'WO') THEN
      OGD = 1
   ELSE IF (LOGD .EQ. '05') THEN
      OGD = 5
   ELSE IF (LOGD .EQ. '06') THEN
      OGD = 6
   END IF
   WRITE (2,4) E1NUM,BOD,PRI,OGD,LOEXP,PMOS,
     C             OT,AMOS1,AMOS2,UR,SEX,MAC
   GOTO 30
 4 FORMAT (I4,1X,A9,1X,A1,1X,I1,1X,A1,1X,I4,1X,
     C       A8,1X,I4,1X,I4,1X,A1,1X,A1,1X,A8)
 7 FORMAT (I4,1X,I3)
200 STOP
   END
```

```
* THIS PROGRAM CLASSIFIES EACH ENTRY IN THE E1 CARDS
* BY ITS GROUPING CATEGORY AND ITS MOS CATEGORY.
* THE DATA FOR THE GROUPING COMES FROM
* THE 'GROUP CRIT' FILE.  THE MOS CATEGORIES ARE:
*   1  -  PMOS AND 2 AMOS
*   2  -  PMOS AND 1 AMOS
*   3  -  PMOS ONLY
*   4  -  OCC FIELD/SUBOCC FIELD
*   5  -  OFFICER TYPE
*   6  -  AMOS ONLY
*
      PROGRAM GRE1
      CHARACTER*1 EXP,SEX,UR,EXPANS,SEXANS,URANS
      CHARACTER*1 MORE,SMORE,B1(4),B2(4),V(5),CHVAL,PRI
      CHARACTER*1 SV(128,6),BLANK1,BLANK2
      CHARACTER*2 CGROUP,PGROUP
      CHARACTER*5 TANS
      CHARACTER*6 CRNAM1, CRNAM2
      CHARACTER*8 OT,MAC
      CHARACTER*9 BOD
      INTEGER GD,PMOS,AMOS1,AMOS2,E1NUM,JMOS
      INTEGER I, J, K, L, M, N, CHNUM1, CHNUM2
      INTEGER NC(5),C1,C2,C3,C4,C5
      OPEN (UNIT = 1, FILE = 'GROUP CRIT A', STATUS = 'OLD')
      OPEN (UNIT = 2, FILE = 'EXPANDED E1 A', STATUS = 'OLD')
      OPEN (UNIT = 3, FILE = 'GROUPE1 FILE A', STATUS = 'NEW')
      OPEN (UNIT = 4, FILE = 'PGROUP ARRAY A', STATUS = 'NEW')
      JMOS = 0
* SET THE STANDARD VALUES
      SV(2,2) = '0'
      SV(2,3) = '1'
      SV(1,2) = '1'
      SV(1,3) = '2'
      SV(3,2) = 'M'
      SV(3,3) = 'F'
      SV(1,1) = '*'
      SV(2,1) = '*'
      SV(3,1) = '*'
      C1 = 0
      C2 = 0
      C3 = 0
      C4 = 0
      C5 = 0
* READ IN DATA FROM GROUP CRIT FILE
      READ(1,2) EXPANS
      READ(1,2) URANS
      READ(1,2) SEXANS
      TANS = 'NNNNN'
      TANS(1:1) = EXPANS
      TANS(2:2) = URANS
      TANS(3:3) = SEXANS
      N = 0
      IF (EXPANS .EQ. 'Y') N = N + 1
      IF (URANS .EQ. 'Y') N = N + 1
      IF (SEXANS .EQ. 'Y') N = N + 1
      READ(1,2) MORE
      TANS(4:4) = MORE
      IF (MORE .EQ. 'Y') N = N + 1
* CHECK FOR BLANK1 DATA
      IF (MORE .EQ. 'Y') THEN
         READ(1,3) CRNAM1
         READ(1,5) CHNUM1
         DO 15 J = 1,CHNUM1
            READ(1,4) CHVAL
            B1(J) = CHVAL
   15    CONTINUE
         READ(1,2) SMORE
         TANS(5:5) = SMORE
```

```
         IF (SMORE .EQ. 'Y') N = N + 1
    2    FORMAT(A1)
* CHECK FOR BLANK2 DATA
         IF (SMORE .EQ. 'Y') THEN
            READ(1,3) CRNAM2
    3       FORMAT(A6)
            READ(1,5) CHNUM2
            DO 20 J = 1,CHNUM2
               READ(1,4) CHVAL
    4          FORMAT(A1)
    5          FORMAT(I1)
               B2(J) = CHVAL
   20       CONTINUE
         END IF
      END IF
      WRITE(6,*) 'THE VALUE OF N = ',N
* MAKE THE BASIC VALUE ARRAY AND SET GROUPS
      IF (TANS(4:5) .NE. 'NN') GOTO 250
         IF (EXPANS .EQ. 'Y') THEN
            C1 = 1
            IF (URANS .EQ. 'Y') THEN
               C2 = 2
               IF (SEXANS .EQ. 'Y') C3 = 3
            ELSE
               IF (SEXANS .EQ. 'Y') C2 = 3
            ENDIF
         ELSE
            IF (URANS .EQ. 'Y') THEN
               C1 = 2
               IF (SEXANS .EQ. 'Y') C2 = 3
            ELSE
               IF (SEXANS .EQ. 'Y') C1 = 3
            ENDIF
          END IF
          WRITE (6,*) 'C1 = ',C1
          WRITE (6,*) 'C2 = ',C2
          WRITE (6,*) 'C3 = ',C3
          WRITE (6,*) 'C4 = ',C4
          WRITE (6,*) 'C5 = ',C5
* SET GROUPS FOR BASIC THREE
          IF (N .EQ. 1) THEN
   70        READ(2,11,END = 200) E1NUM,BOD,PRI,GD,V(2),
     C                     PMOS,OT,AMOS1,AMOS2,V(1),V(3),MAC
             IF (V(C1) .EQ. SV(C1,1)) THEN
                CGROUP = '0'
             ELSE IF (V(C1) .EQ. SV(C1,2)) THEN
                CGROUP = '1'
             ELSE
                CGROUP = '2'
             END IF
* CHECK THE PMOS/AMOS/OT
             IF (PMOS .LT. 169 .AND. AMOS1 .NE. 0 .AND.
     C                AMOS2 .NE. 0) THEN
                PGROUP = '1'
             ELSE IF (PMOS .LT. 169 .AND. ((AMOS1 .EQ. 0
     C             .AND. AMOS2 .NE. 0) .OR. (AMOS1 .NE. 0)
     C             .AND. (AMOS2 .EQ. 0))) THEN
                 PGROUP = '2'
             ELSE IF (PMOS .LT. 169) THEN
                PGROUP = '3'
             ELSE IF (PMOS .GT. 1000)THEN
                PGROUP = '4'
             ELSE IF (PMOS .EQ. 239) THEN
                PGROUP = '5'
             ELSE IF (PMOS .EQ. 240) THEN
                PGROUP = '6'
             END IF
             WRITE(3,12) E1NUM,BOD,PRI,GD,V(2),PMOS,OT,
     C                   AMOS1,AMOS2,JMOS,V(1),V(3),
     C                   MAC,CGROUP,PGROUP
```

```
            GOTO 70
         ELSE IF (N .EQ. 2) THEN
80          READ(2,11,END = 200) E1NUM,BOD,PRI,GD,V(2),
     C                           PMOS,OT,AMOS1,
     C                           AMOS2,V(1),V(3),MAC
            IF (V(C1) .EQ. SV(C1,1)) THEN
               IF (V(C2) .EQ. SV(C2,1)) THEN
                  CGROUP = '0'
               ELSE IF (V(C2) .EQ. SV(C2,2)) THEN
                  CGROUP = '1'
               ELSE
                  CGROUP = '2'
               END IF
            ELSE IF (V(C1) .EQ. SV(C1,2)) THEN
               IF (V(C2) .EQ. SV(C2,1)) THEN
                  CGROUP = '3'
               ELSE IF (V(C2) .EQ. SV(C2,2)) THEN
                  CGROUP = '4'
               ELSE
                  CGROUP = '5'
               END IF
            ELSE
               IF (V(C2) .EQ. SV(C2,1)) THEN
                  CGROUP = '6'
               ELSE IF (V(C2) .EQ. SV(C2,2)) THEN
                  CGROUP = '7'
               ELSE
                  CGROUP = '8'
               END IF
            END IF
* CHECK THE PMOS/AMOS/OT
            IF (PMOS .LT. 169 .AND. AMOS1 .NE. 0 .AND.
     C           AMOS2 .NE. 0) THEN
               PGROUP = '1'
            ELSE IF (PMOS .LT. 169 .AND. ((AMOS1 .EQ. 0
     C        .AND. AMOS2 .NE. 0) .OR. (AMOS1 .NE. 0)
     C        .AND. (AMOS2 .EQ. 0))) THEN
               PGROUP = '2'
            ELSE IF (PMOS .LT. 169) THEN
               PGROUP = '3'
            ELSE IF (PMOS .GT. 1000)THEN
               PGROUP = '4'
            ELSE IF (PMOS .EQ. 239) THEN
               PGROUP = '5'
            ELSE IF (PMOS .EQ. 240) THEN
               PGROUP = '6'
            END IF
            WRITE(3,12) BOD,PRI,GD,V(2),PMOS,OT,
     C                  AMOS1,AMOS2,JMOS,V(1),V(3),MAC,
     C                  MAC,CGROUP,PGROUP
            GOTO 80
         ELSE
            WRITE(6,*) 'INSIDE N = 3 LOOP'
90          READ(2,11,END = 200) E1NUM,BOD,PRI,GD,V(2),
     C                           PMOS,OT,AMOS1,
     C                           AMOS2,V(1),V(3),MAC
            IF (V(C1) .EQ. SV(C1,1)) THEN
               IF (V(C2) .EQ. SV(C2,1)) THEN
                  IF (V(C3) .EQ. SV(C3,1)) THEN
                     CGROUP = '0'
                  ELSE IF (V(C3) .EQ. SV(C3,2)) THEN
                     CGROUP = '1'
                  ELSE
                     CGROUP = '2'
                  END IF
               ELSE IF (V(C2) .EQ. SV(C2,2)) THEN
                  IF (V(C3) .EQ. SV(C3,1)) THEN
                     CGROUP = '3'
                  ELSE IF (V(C3) .EQ. SV(C3,2)) THEN
                     CGROUP = '4'
```

```
                  ELSE
                     CGROUP = '5'
                  END IF
               ELSE
                  IF (V(C3) .EQ. SV(C3,1)) THEN
                     CGROUP = '6'
                  ELSE IF (V(C3) .EQ. SV(C3,2)) THEN
                     CGROUP = '7'
                  ELSE
                     CGROUP = '8'
                  END IF
               END IF
            ELSE IF (V(C1) .EQ. SV(C1,2)) THEN
               IF (V(C2) .EQ. SV(C2,1)) THEN
                  IF (V(C3) .EQ. SV(C3,1)) THEN
                     CGROUP = '9'
                  ELSE IF (V(C3) .EQ. SV(C3,2)) THEN
                     CGROUP = '10'
                  ELSE
                     CGROUP = '11'
                  END IF
               ELSE IF (V(C2) .EQ. SV(C2,2)) THEN
                  IF (V(C3) .EQ. SV(C3,1)) THEN
                     CGROUP = '12'
                  ELSE IF (V(C3) .EQ. SV(C3,2)) THEN
                     CGROUP = '13'
                  ELSE
                     CGROUP = '14'
                  END IF
               ELSE
                  IF (V(C3) .EQ. SV(C3,1)) THEN
                     CGROUP = '15'
                  ELSE IF (V(C3) .EQ. SV(C3,2)) THEN
                     CGROUP = '16'
                  ELSE
                     CGROUP = '17'
                  END IF
               END IF
            ELSE
               IF (V(C2) .EQ. SV(C2,1)) THEN
                  IF (V(C3) .EQ. SV(C3,1)) THEN
                     CGROUP = '18'
                  ELSE IF (V(C3) .EQ. SV(C3,2)) THEN
                     CGROUP = '19'
                  ELSE
                     CGROUP = '20'
                  END IF
               ELSE IF (V(C2) .EQ. SV(C2,2)) THEN
                  IF (V(C3) .EQ. SV(C3,1)) THEN
                     CGROUP = '21'
                  ELSE IF (V(C3) .EQ. SV(C3,2)) THEN
                     CGROUP = '22'
                  ELSE
                     CGROUP = '23'
                  END IF
               ELSE
                  IF (V(C3) .EQ. SV(C3,1)) THEN
                     CGROUP = '24'
                  ELSE IF (V(C3) .EQ. SV(C3,2)) THEN
                     CGROUP = '25'
                  ELSE
                     CGROUP = '26'
                  END IF
               END IF
            END IF
C CHECK THE PMOS/AMOS/OT
            IF (PMOS .LT. 169 .AND. AMOS1 .NE. 0 .AND.
     C             AMOS2 .NE. 0) THEN
               PGROUP = '1'
            ELSE IF (PMOS .LT. 169 .AND. ((AMOS1 .EQ. 0
```

```
     C            .AND. AMOS2 .NE. 0) .OR. (AMOS1 .NE. 0)
     C            .AND. (AMOS2 .EQ. 0))) THEN
                 PGROUP = '2'
               ELSE IF (PMOS .LT. 169) THEN
                  PGROUP = '3'
               ELSE IF (PMOS .GT. 1000)THEN
                  PGROUP = '4'
               ELSE IF (PMOS .EQ. 239) THEN
                  PGROUP = '5'
               ELSE IF (PMOS .EQ. 240) THEN
                  PGROUP = '6'
               END IF
               WRITE(3,12) E1NUM,BOD,PRI,GD,V(2),PMOS,OT,
     C                     AMOS1,AMOS2,JMOS,V(1),V(3),
     C                     MAC,CGROUP,PGROUP
               GOTO 90
               END IF
C MAKE ARRAY FOR PGROUP ARRAY FOR MATCHING ROUTINE
  200 WRITE(4,9) N
      IF (N .EQ. 1) THEN
         WRITE(4,9) C1
         DO 100 I = 1,3
            WRITE(4,6) SV(C1,I)
  100    CONTINUE
      ELSE IF (N .EQ. 2) THEN
         WRITE(4,9) C1
         WRITE(4,9) C2
         DO 120 I = 1,3
            DO 110 J = 1,3
               WRITE(4,7) SV(C1,I), SV(C2,J)
  110       CONTINUE
  120    CONTINUE
      ELSE IF (N .EQ. 3) THEN
         WRITE(4,9) C1
         WRITE(4,9) C2
         WRITE(4,9) C3
         DO 150 I = 1,3
            DO 140 J = 1,3
               DO 130 K = 1,3
                WRITE (4,8) SV(C1,I), SV(C2,J), SV(C3,K)
  130          CONTINUE
  140       CONTINUE
  150    CONTINUE
      END IF
    6 FORMAT (A1)
    7 FORMAT (2A1)
    8 FORMAT (A1,1X,A1,1X,A1)
    9 FORMAT (I1)
   11       FORMAT(I4,1X,A9,1X,A1,1X,I1,1X,A1,1X,I4,1X,
     C              A8,1X,I4,1X,I4,1X,A1,1X,A1,1X,A8)
   12 FORMAT(I4,1X,A9,1X,A1,1X,I1,1X,A1,1X,I4,1X,
     C       A8,1X,I4,1X,I4,1X,I4,1X,A1,1X,A1,1X,A8,1X,
     C       A2,1X,A2)
  250 STOP
      END
```

EXPASR FORTRAN

```
* THIS PROGRAM EXPANDS THE ASR INTO SINGLE LINE FORMAT
      PROGRAM EXPASR
      CHARACTER*3 MCC
      INTEGER GEN,COL,LTCOL,MAJ,CAPT,LT,WO,ASRNUM,GD
      INTEGER MOS(1:9999),INMOS,MOSNUM,BMOS,TMOS
      DATA MOS / 9999*0/
      OPEN (UNIT = 1, FILE = 'FULL ASR A', STATUS = 'OLD')
      OPEN (UNIT = 2, FILE = 'EXP ASR A', STATUS = 'NEW')
      OPEN (UNIT = 3, FILE = 'BMOS B1 A', STATUS = 'OLD')
* READ IN THE MOS'S FROM BMOS B1 AND MAKE THE MOS ARRAY
    5 READ(3,6,END = 10) INMOS,MOSNUM
      MOS(INMOS) = MOSNUM
      GOTO 5
* READ IN A LINE FROM FULL ASR, BREAK INTO SEPARATE LINES
   10 READ(1,3,END = 200) BMOS,MCC,GEN,COL,LTCOL,
     C                    MAJ,CAPT,LT,WO
    3 FORMAT(I4,1X,A3,10X,I4,2X,I4,2X,I4,2X,I4,2X,
             I4,2X,I4,2X,I4)
* CHECK TO SEE THAT THE MOS IS A VALID ONE
      IF (MOS(BMOS) .EQ. 0) GOTO 10
* MOS IS OK - CONVERT ASR LINE TO SINGLE LINE FORMAT
* ALSO CREATE MG ARRAY
      TMOS = MOS(BMOS)
      IF (GEN .NE. 0) THEN
         GD = 7
         WRITE(2,4) TMOS,GD,MCC,GEN
      END IF
      IF (COL .NE. 0) THEN
         GD = 6
         WRITE(2,4) TMOS,GD,MCC,COL
      END IF
      IF (LTCOL .NE. 0) THEN
         GD = 5
         WRITE(2,4) TMOS,GD,MCC,LTCOL
      END IF
      IF (MAJ .NE. 0) THEN
         GD = 4
         WRITE(2,4) TMOS,GD,MCC,MAJ
      END IF
      IF (CAPT .NE. 0) THEN
         GD = 3
         WRITE(2,4) TMOS,GD,MCC,CAPT
      END IF
      IF (LT .NE. 0) THEN
         GD = 2
         WRITE(2,4) TMOS,GD,MCC,LT
      END IF
      IF (WO .NE. 0) THEN
         GD = 1
         WRITE(2,4) TMOS,GD,MCC,WO
      END IF
      GOTO 10
    4 FORMAT(I4,1X,I1,1X,A3,1X,I4)
    6 FORMAT(I4,1X,I3)
  200 STOP
      END
```

EXPC1 SAS

```
*THIS PROGRAM PUTS THE C1 CARDS IN SINGLE LINE FORMAT;
CMS FILEDEF FONE DISK FULL C1 C;
CMS FILEDEF FTWO DISK EXPC1 FILE C(RECFM F
    LRECL 80 BLOCK 80;
*READ IN THE C1 CARDS AND PUT IN SINGLE LINE FORMAT;
DATA DONE;
  INFILE FONE;
  INPUT MOS $ MCC $ GD ADJ $ @@;
  ACT = SUBSTR(ADJ,1,1);
  AMT = SUBSTR(ADJ,2,4);
  DROP ADJ;
  FILE FTWO;
  PUT MOS 1-4 MCC 6-8 GD 10 ACT 12 AMT 14-17;
```

ASORT SAS

```
* THIS PROGRAM SORTS EXPASR INTO MOS GD MCC FORMAT;
CMS FILEDEF FONE DISK EXP ASR A;
CMS FILEDEF FTWO DISK SORTED ASR A(RECFM F
    LRECL 80 BLOCK 80;
DATA DONE;
   INFILE FONE;
   INPUT MOS 1-4 GD  6 MCC $ 8-10 AMT $ 12-15;
PROC SORT;
   BY MOS GD MCC;
DATA _NULL_;
   SET DONE;
   FILE FTWO;
   PUT MOS 1-4 GD 6 MCC 8-10 AMT 12-15;
```

CSORT SAS

```
* THIS PROGRAM SORTS THE EXPC1 FILE INTO MOS GD MCC FORMAT;
CMS FILEDEF FONE DISK EXPC1 FILE A;
CMS FILEDEF FTWO DISK EXP C1 A(RECFM F LRECL 80 BLOCK 80;
DATA DONE;
   INFILE FONE;
   INPUT MOS $ 1-4 MCC $ 6-8 GD $ 10 ACT $ 12 AMT $ 14-17;
PROC SORT;
   BY MOS GD MCC;
DATA _NULL_;
   SET DONE;
   FILE FTWO;
   PUT MOS 1-4 MCC 6-8 GD 10 ACT 12 AMT 14-17;
```

ROLLC1 FORTRAN

```
* THIS PROGRAM ROLLS UP ANY DUPLICATE C1 LINES
* IN THE C1 FILE
      PROGRAM ROLLC1
      CHARACTER*1 CACT,PACT
      CHARACTER*3 CMCC,PMCC
      INTEGER CAMT,PAMT,FLAG,INMOS,MOSNUM
      INTEGER CMOS,PMOS,CGD,PGD,MOS(1:9999)
      DATA MOS / 9999*0 /
      OPEN (UNIT = 1, FILE = 'EXP C1 A', STATUS = 'OLD')
      OPEN (UNIT = 2, FILE = 'ROLLED C1 A', STATUS = 'NEW')
      OPEN (UNIT = 3, FILE = 'BMOS B1 A', STATUS = 'OLD')
      FLAG = 0
* READ IN THE MOS'S FROM THE BMOS B1 AND MAKE THE MOS ARRAY
    5 READ(3,6,END = 8)INMOS,MOSNUM
      MOS(INMOS) = MOSNUM
      GOTO 5
    8 READ(1,3,END = 200) PMOS,PMCC,PGD,PACT,PAMT
* READ IN A LINE AND COMPARE IT WITH THE PREVIOUS LINE
* IF IT IS THE SAME, THEN ROLL UP THE TWO
   10 READ(1,3,END = 200) CMOS,CMCC,CGD,CACT,CAMT
      IF ((PMOS .EQ. CMOS) .AND. (PGD .EQ. CGD) .AND.
     C    (PMCC .EQ. CMCC)) THEN
          FLAG = 1
          IF ((PACT .NE. 'S') .AND. (CACT .NE. 'S')) THEN
             PAMT = PAMT + CAMT
          ELSE IF (PACT .NE. 'S' .AND. CACT .EQ. 'S') THEN
             PAMT = PAMT - CAMT
          ELSE
             PAMT = CAMT - PAMT
          END IF
          GOTO 10
      ELSE
          FLAG = 0
          WRITE(2,3) MOS(PMOS),PMCC,PGD,PACT,PAMT
          PMOS = CMOS
          PMCC = CMCC
          PGD = CGD
          PACT = CACT
          PAMT = CAMT
          GOTO 10
      END IF
  200 IF (FLAG .EQ. 1) THEN
          WRITE(2,3) MOS(PMOS),PMCC,PGD,PACT,PAMT
      END IF
      WRITE(2,3) MOS(PMOS),PMCC,PGD,PACT,PAMT
    3 FORMAT(I4,1X,A3,1X,I1,1X,A1,1X,I4)
    6 FORMAT(I4,1X,I3)
      STOP
      END
```

ADJASR FORTRAN

```
* THIS PROGRAM MODIFIES THE ASR IN ACCORDANCE WITH
* THE INFORMATION CONTAINED IN THE C1 CARDS.
* THE C1 CARDS MAY ADD NEW ASR LINES,
* EFFECTIVELY ELIMINATE ASR LINES OR MODIFY THE NUMBER
* OF PERSONNEL FOR A PARTICULAR ASR LINE.
*
      PROGRAM ADJASR
      CHARACTER*1 CACT
      CHARACTER*3 CMCC,AMCC
      INTEGER CAMT,ASRAMT
      INTEGER AGD,CGD,CMOS,AMOS
      OPEN (UNIT = 1, FILE = 'ROLLED C1 A', STATUS = 'OLD')
      OPEN (UNIT = 2, FILE = 'SORTED ASR A', STATUS = 'OLD')
      OPEN (UNIT = 4, FILE = 'ADJUSTED ASR A', STATUS = 'NEW')
      READ(1,5,END = 200) CMOS,CMCC,CGD,CACT,CAMT
   10 READ(2,6,END = 200) AMOS,AGD,AMCC,ASRAMT
   20 IF (CMOS .EQ. AMOS) THEN
         IF (CGD .EQ. AGD) THEN
            IF (CMCC .EQ. AMCC) THEN
               IF (CACT .EQ. 'E') ASRAMT = CAMT
               IF (CACT .EQ. 'A') ASRAMT = ASRAMT + CAMT
               IF (CACT .EQ. 'S') ASRAMT = ASRAMT - CAMT
               IF (ASRAMT .GT. 0) THEN
                  WRITE(4,6) AMOS,AGD,AMCC,ASRAMT
               END IF
               READ(1,5,END = 200) CMOS,CMCC,CGD,CACT,CAMT
               GOTO 10
            ELSE IF (CMCC .GT. AMCC) THEN
               WRITE(4,6) AMOS,AGD,AMCC,ASRAMT
               GOTO 10
            ELSE
               IF (CACT .NE. 'S' .AND. CAMT .NE. 0) THEN
                  WRITE(4,6) CMOS,CGD,CMCC,CAMT
               END IF
               READ(1,5,END = 200) CMOS,CMCC,CGD,CACT,CAMT
               GOTO 20
            END IF
         ELSE IF (CGD .GT. AGD) THEN
            WRITE(4,6) AMOS,AGD,AMCC,ASRAMT
            GOTO 10
         ELSE
            READ(1,5,END = 200) CMOS,CMCC,CGD,CACT,CAMT
            GOTO 20
         END IF
      ELSE IF (CMOS .GT. AMOS) THEN
         WRITE(4,6) AMOS,AGD,AMCC,ASRAMT
         GOTO 10
      ELSE
         READ(1,5,END = 200) CMOS,CMCC,CGD,CACT,CAMT
         GOTO 20
      END IF
    5 FORMAT(I4,1X,A3,1X,I1,1X,A1,1X,I4)
    6 FORMAT(I4,1X,I1,1X,A3,1X,I4)
  200 STOP
      END
```

NUMASR FORTRAN

```
* THIS PROGRAM WILL NUMBER THE ASR AND GENERATE AN ARRAY
* THAT POINTS TO EACH MOS/GRADE STARTING LINE
      PROGRAM NUMASR
      CHARACTER*3 MCC
      INTEGER BMOS,CAP,ASRNUM,GD,PMOS,PGD,MG(1:232,1:7)
      DATA MG / 1624*0 /
      OPEN (UNIT = 1, FILE = 'ADJUSTED ASR A', STATUS = 'OLD')
      OPEN (UNIT = 2, FILE = 'MG ARRAY A', STATUS = 'NEW')
      OPEN (UNIT = 3, FILE = 'FINAL ASR A', STATUS = 'NEW')
      ASRNUM = 1
      PMOS = 0
      PGD = 0
* READ IN A LINE FROM THE ASR AND NUMBER IT
   10 READ(1,4,END = 200) BMOS,GD,MCC,CAP
      IF (PMOS .NE. BMOS .OR. PGD .NE. GD) THEN
         MG(BMOS,GD) = ASRNUM
      END IF
      WRITE(3,5) ASRNUM,BMOS,GD,MCC,CAP
      ASRNUM = ASRNUM + 1
      PMOS = BMOS
      PGD = GD
      GOTO 10
  200 DO 205 I = 1,232
         WRITE(2,6) MG(I,1),MG(I,2),MG(I,3),MG(I,4),
     C              MG(I,5),MG(I,6),MG(I,7)
  205 CONTINUE
    4 FORMAT(I4,1X,I1,1X,A3,1X,I4)
    5 FORMAT(I4,1X,I4,1X,I1,1X,A3,1X,I4)
    6 FORMAT(I4,1X,I4,1X,I4,1X,I4,1X,I4,1X,I4,1X,I4)
      STOP
      END
```

E2SORT SAS

```
* THIS PROGRAM NUMBERS AND SORTS THE E2 CARDS;
* BY MOS,GD,MCC:
CMS FILEDEF FONE DISK FULL E2 A;
CMS FILEDEF FTWO DISK SORT E2 A(RECFM F LRECL 80 BLOCK 80;
DATA DONE;
   RETAIN E2NUM 0;
   INFILE FONE;
   INPUT MOS $ 1-4 MCC $ 17-19 GD $ 21-22 SAMT $ 24-27
         SPL $ 29 BOD $ 34-42 MAC $ 45-52;
   E2NUM + 1;
PROC SORT;
   BY MOS GD MCC SPL BOD MAC;
DATA _NULL_;
   SET DONE;
   FILE FTWO;
   PUT E2NUM 1-4 MOS 6-9 MCC 11-13 GD 15-16 SAMT $ 18-21
       SPL $ 23 BOD $ 25-33 MAC $ 35-42;
```

```
* THIS PROGRAM ROLLS THE E2 FILE INTO BMIS SETS.
* A BMIS IS ANY SET OF
* E2 CARDS WITH THE SAME MOS,GD,MCC,SPL,BOD,& MAC.
      PROGRAM BMIS
      CHARACTER*1 SPL,TSPL
      CHARACTER*2 GD,TGD
      CHARACTER*3 MCC,TMCC
      CHARACTER*4 MOS,E2NUM,TMOS,TE2NUM,SAMT,TSAMT
      CHARACTER*8 MAC,TMAC
      CHARACTER*9 BOD,TBOD
      INTEGER BNUM
      OPEN (UNIT = 1, FILE = 'SORT E2 A', STATUS = 'OLD')
      OPEN (UNIT = 2, FILE = 'BMIS E2 A', STATUS = 'NEW')
      BNUM = 1
      READ(1,3,END = 200) E2NUM,MOS,MCC,GD,SAMT,SPL,BOD,MAC
      TE2NUM = E2NUM
      TMOS = MOS
      TMCC = MCC
      TGD = GD
      TSAMT = SAMT
      TSPL = SPL
      TBOD = BOD
      TMAC = MAC
   10 READ(1,3,END = 200) E2NUM,MOS,MCC,GD,SAMT,SPL,BOD,MAC
* CHECK TO SEE IF THE TWO E2 LINES ARE IN THE SAME BMIS
      IF (TMOS .EQ. MOS .AND. TMCC .EQ. MCC .AND.
     C    TGD .EQ. GD .AND. TSPL .EQ. SPL .AND.
     C    TBOD .EQ. BOD .AND. TMAC .EQ. MAC) THEN
          WRITE(2,4) TE2NUM,TMOS,TMCC,TGD,TSAMT,TSPL,
     C               TBOD,TMAC,BNUM
          GOTO 10
       ELSE
          WRITE(2,4) TE2NUM,TMOS,TMCC,TGD,TSAMT,TSPL,
     C               TBOD,TMAC,BNUM
          BNUM = BNUM + 1
          TE2NUM = E2NUM
          TMOS = MOS
          TMCC = MCC
          TGD = GD
          TSAMT = SAMT
          TSPL = SPL
          TBOD = BOD
          TMAC = MAC
          GOTO 10
       END IF
  200 WRITE(2,4) TE2NUM,TMOS,TMCC,TGD,TSAMT,TSPL,
     C            TBOD,TMAC,BNUM
    3 FORMAT(A4,1X,A4,1X,A3,1X,A2,1X,A4,1X,A1,1X,A9,1X,A8)
    4 FORMAT(A4,1X,A4,1X,A3,1X,A2,1X,A4,1X,A1,1X,A9,1X,
     C       A8,1X,I4)
       STOP
       END
```

```
* THIS PROGRAM SPLITS THE E2 FILE INTO SPL'S
* AND CONVERTS THE FORMAT
      PROGRAM REND
      CHARACTER*1 SPL
      CHARACTER*2 INGD
      CHARACTER*3 MCC
      CHARACTER*4 SAMT
      CHARACTER*8 MAC
      CHARACTER*9 BOD
      INTEGER E2NUM,AMOS(1:9999),INMOS,MOSNUM,GD,MOS,BMIS
      DATA AMOS / 9999*0 /
      OPEN (UNIT = 1, FILE = 'BMIS E2 A', STATUS = 'OLD')
      OPEN (UNIT = 2, FILE = 'SPL2 E2 A', STATUS = 'NEW')
      OPEN (UNIT = 3, FILE = 'SPL3 E2 A', STATUS = 'NEW')
      OPEN (UNIT = 4, FILE = 'SPL4 E2 A', STATUS = 'NEW')
      OPEN (UNIT = 7, FILE = 'SPL5 E2 A', STATUS = 'NEW')
      OPEN (UNIT = 8, FILE = 'BMOS B1 A', STATUS = 'OLD')
* READ IN THE MOS'S FROM BMOS B1 AND MAKE THE ARRAY
    5 READ (8,3,END = 10) INMOS,MOSNUM
      AMOS(INMOS) = MOSNUM
      GOTO 5
   10 READ (1,4,END = 200) E2NUM,MOS,MCC,INGD,SAMT,SPL,
     C                     BOD,MAC,BMIS
      IF (INGD .EQ. '03') THEN
         GD = 3
      ELSE IF (INGD .EQ. '02') THEN
         GD = 2
      ELSE IF (INGD .EQ. '04') THEN
         GD = 4
      ELSE IF (INGD .EQ. '05') THEN
         GD = 5
      ELSE IF (INGD .EQ. 'WO') THEN
         GD = 1
      ELSE IF (INGD .EQ. '06') THEN
         GD = 6
      END IF
      IF (SPL .EQ. '2') THEN
         WRITE(2,2) E2NUM,AMOS(MOS),MCC,GD,SAMT,SPL,
     C              BOD,MAC,BMIS
      ELSE IF (SPL .EQ. '3') THEN
         WRITE(3,2) E2NUM,AMOS(MOS),MCC,GD,SAMT,SPL,
     C              BOD,MAC,BMIS
      ELSE IF (SPL .EQ. '4') THEN
         WRITE(4,2) E2NUM,AMOS(MOS),MCC,GD,SAMT,SPL,
     C              BOD,MAC,BMIS
      ELSE
         WRITE(7,2) E2NUM,AMOS(MOS),MCC,GD,SAMT,SPL,
     C              BOD,MAC,BMIS
      END IF
      GOTO 10
    2 FORMAT(I4,1X,I4,1X,A3,1X,I1,1X,A4,1X,A1,1X,
     C       A9,1X,A8,1X,I4)
    4 FORMAT(I4,1X,I4,1X,A3,1X,A2,1X,A4,1X,A1,1X,
     C       A9,1X,A8,1X,I4)
    3 FORMAT(I4,1X,I3)
  200 STOP
      END
```

```
* THIS PROGRAM MERGES THE SPL2 E2 CARDS WITH THE ASR
* SPL2 IS A REND FROM THE BMIS FILE
* MG ARRAY IS A POINTER ARRAY INTO THE ASR DATA
*
      PROGRAM BMIX
      CHARACTER*1 SPL,AP,MCC1,C1,C2,C3
      CHARACTER*2 MCC2
      CHARACTER*3 EMCC,AMCC,TMCC(8000)
      CHARACTER*8 MAC
      CHARACTER*9 BOD
      INTEGER EMOS,AMOS,TMOS(8000),EGD,AGD,TGD(8000)
      INTEGER E2NUM,ASRNUM,CAP,TNUM(8000),TCAP(8000),I
      INTEGER SAMT,MG(1:232,1:7),N,BMIS,F,HOLD
      DATA MG / 1624*0 /
      OPEN (UNIT = 1, FILE = 'SPL2 E2 A', STATUS = 'OLD')
      OPEN (UNIT = 2, FILE = 'SPL3 E2 A', STATUS = 'OLD')
      OPEN (UNIT = 3, FILE = 'SPL4 E2 A', STATUS = 'OLD')
      OPEN (UNIT = 4, FILE = 'SPL5 E2 A', STATUS = 'OLD')
      OPEN (UNIT = 7, FILE = 'TOTAL MIX A', STATUS = 'NEW')
      OPEN (UNIT = 8, FILE = 'FINAL ASR A', STATUS = 'OLD')
      OPEN (UNIT = 9, FILE = 'MG ARRAY A', STATUS = 'OLD')
      OPEN (UNIT = 10, FILE = 'UNUSED ASR A', STATUS = 'NEW')
      I = 0
* READ DATA INTO MG ARRAY
      DO 8 N = 1,232
         READ(9,7,END = 10) MG(N,1),MG(N,2),MG(N,3),
     C                  MG(N,4), MG(N,5),MG(N,6),MG(N,7)
    8 CONTINUE
* READ ASR INTO AN ARRAY
   10 READ (8,2,END = 20) ASRNUM,AMOS,AGD,AMCC,CAP
      I = I + 1
      TNUM(I) = ASRNUM
      TMOS(I) = AMOS
      TGD(I) = AGD
      TMCC(I) = AMCC
      TCAP(I) = CAP
      GOTO 10
   20 HOLD = I
      I = 1
* READ IN E2 AND DETERMINE WHAT STAR IT IS
      DO 900 F = 1,4
   25 READ (F,3,END = 900) E2NUM,EMOS,EMCC,EGD,AP,SAMT,SPL,
     C                     BOD,MAC,BMIS
      I = MG(EMOS,EGD)
      IF (I .EQ. 0) GOTO 25
      C1 = EMCC(1:1)
      C2 = EMCC(2:2)
      C3 = EMCC(3:3)
      IF (C1 .EQ. '*') THEN
         GOTO 200
      ELSE IF (C2 .EQ. '*') THEN
         MCC1 = EMCC(1:1)
         GOTO 300
      ELSE IF (C3 .EQ. '*') THEN
         MCC2 = EMCC(1:2)
         GOTO 400
      ELSE
         GOTO 100
      END IF
* THIS IS THE NOSTAR SUBSECTION
  100 IF (TMOS(I) .EQ. EMOS) THEN
        IF (TGD(I) .EQ. EGD) THEN
          IF (TMCC(I) .EQ. EMCC) THEN
            IF (AP .EQ. 'A' .AND. TCAP(I) .GE. SAMT) THEN
               TCAP(I) = TCAP(I) - SAMT
               WRITE (7,4) E2NUM,TNUM(I),EMOS,EGD,EMCC,
     C                           SAMT,SPL,BOD,BMIS
             ELSE
```

```
                IF (TCAP(I) .NE. 0) THEN
                   WRITE(7,4)E2NUM,TNUM(I),EMOS,EGD,EMCC,
                             TCAP(I),SPL,BOD,BMIS
                END IF
                TCAP(I) = 0
             ENDIF
             GOTO 25
          ELSE IF (TMCC(I) .LT. EMCC) THEN
             I = I + 1
             IF (I .GT. HOLD) THEN
                GOTO 25
             ELSE
                GOTO 100
             END IF
          ELSE
             GOTO 25
          ENDIF
       ELSE IF (TGD(I) .LT. EGD) THEN
          I = I + 1
          IF (I .GT. HOLD) THEN
             GOTO 25
          ELSE
             GOTO 100
          END IF
       ELSE
          GOTO 25
       ENDIF
    ELSE IF (TMOS(I) .LT. EMOS) THEN
       I = I + 1
       IF (I .GT. HOLD) THEN
          GOTO 25
       ELSE
          GOTO 100
       END IF
    ELSE
       GOTO 25
    ENDIF
* THIS IS THE TRESTAR SUBSECTION
  200 IF (TMOS(I) .EQ. EMOS) THEN
         IF (TGD(I) .EQ. EGD) THEN
            IF (AP .EQ. 'A' .AND. TCAP(I) .GE. SAMT) THEN
               TCAP(I) = TCAP(I) - SAMT
               WRITE (7,4) E2NUM,TNUM(I),EMOS,EGD,TMCC(I),
     C                     SAMT,SPL,BOD,BMIS
            ELSE
               IF (TCAP(I) .NE. 0) THEN
                WRITE(7,4)E2NUM,TNUM(I),EMOS,EGD,TMCC(I),
     C                    TCAP(I),SPL,BOD,BMIS
               END IF
               TCAP(I) = 0
            ENDIF
            I = I + 1
            IF (I .GT. HOLD) THEN
               GOTO 25
            ELSE
                GOTO 200
            END IF
         ELSE IF (TGD(I) .LT. EGD) THEN
            I = I + 1
            IF (I .GT. HOLD) THEN
               GOTO 25
            ELSE
                GOTO 200
            END IF
         ELSE
            GOTO 25
         END IF
       ELSE IF (TMOS(I) .LT. EMOS) THEN
          I = I + 1
          IF (I .GT. HOLD) THEN
```

```
            GOTO 25
         ELSE
            GOTO 200
         END IF
      ELSE
         GOTO 25
      ENDIF
* THIS IS THE TWOSTAR SUBSECTION
  300 IF (TMOS(I) .EQ. EMOS) THEN
        IF (TGD(I) .EQ. EGD) THEN
          IF (TMCC(I)(1:1) .EQ. MCC1) THEN
            IF (AP .EQ. 'A' .AND. TCAP(I) .GE. SAMT) THEN
               TCAP(I) = TCAP(I) - SAMT
               WRITE (7,4) E2NUM,TNUM(I),EMOS,EGD,TMCC(I),
     C                     SAMT,SPL,BOD,BMIS
            ELSE
               IF (TCAP(I) .NE. 0) THEN
                WRITE(7,4)E2NUM,TNUM(I),EMOS,EGD,TMCC(I),
     C                       TCAP(I),SPL,BOD,BMIS
               END IF
               TCAP(I) = 0
            ENDIF
            I = I + 1
            IF (I .GT. HOLD) THEN
               GOTO 25
            ELSE
               GOTO 300
             END IF
          ELSE IF (TMCC(I)(1:1) .LT. MCC1) THEN
            I = I + 1
            IF (I .GT. HOLD) THEN
               GOTO 25
            ELSE
               GOTO 300
            END IF
         ELSE
            GOTO 25
         ENDIF
      ELSE IF (TGD(I) .LT. EGD) THEN
         I = I + 1
         IF (I .GT. HOLD) THEN
            GOTO 25
         ELSE
            GOTO 300
         END IF
      ELSE
         GOTO 25
      ENDIF
   ELSE IF (TMOS(I) .LT. EMOS) THEN
      I = I + 1
      IF (I .GT. HOLD) THEN
         GOTO 25
      ELSE
         GOTO 300
      END IF
   ELSE
      GOTO 25
   ENDIF
* THIS IS THE ONESTAR MIX SUBSECTION
  400 IF (TMOS(I) .EQ. EMOS) THEN
        IF (TGD(I) .EQ. EGD) THEN
          IF (TMCC(I)(1:2) .EQ. MCC2) THEN
            IF (AP .EQ. 'A' .AND. TCAP(I) .GE. SAMT) THEN
               TCAP(I) = TCAP(I) - SAMT
               WRITE (7,4) E2NUM,TNUM(I),EMOS,EGD,TMCC(I),
     C                     SAMT,SPL,BOD,BMIS
            ELSE
               IF (TCAP(I) .NE. 0) THEN
                WRITE(7,4)E2NUM,TNUM(I),EMOS,EGD,TMCC(I),
     C                       TCAP(I),SPL,BOD,BMIS
```

```
            END IF
            TCAP(I) = 0
         ENDIF
         I = I + 1
         IF (I .GT. HOLD) THEN
            GOTO 25
         ELSE
            GOTO 400
         END IF
      ELSE IF (TMCC(I)(1:2) .LT. MCC2) THEN
         I = I + 1
         IF (I .GT. HOLD) THEN
            GOTO 25
         ELSE
            GOTO 400
         END IF
      ELSE
         GOTO 25
      ENDIF
   ELSE IF (TGD(I) .LT. EGD) THEN
      I = I + 1
      IF (I .GT. HOLD) THEN
         GOTO 25
      ELSE
         GOTO 400
      END IF
   ELSE
      GOTO 25
   ENDIF
   ELSE IF (TMOS(I) .LT. EMOS) THEN
      I = I + 1
      IF (I .GT. HOLD) THEN
         GOTO 25
      ELSE
         GOTO 400
      END IF
   ELSE
      GOTO 25
   ENDIF
900 CONTINUE
   DO 910 I = 1,HOLD
      IF (TCAP(I) .GT. 0) THEN
         WRITE(10,2)TNUM(I),TMOS(I),TGD(I),TMCC(I),
  C                 TCAP(I)
      END IF
910 CONTINUE
  2 FORMAT(I4,1X,I4,1X,I1,1X,A3,1X,I4)
  3 FORMAT(I4,1X,I4,1X,A3,1X,I1,1X,A1,I3,1X,A1,1X,
   C       A9,1X,A8,1X,I4)
  4 FORMAT(I4,1X,I4,1X,I4,1X,I1,1X,A3,1X,I3,1X,A1,1X,
   C       A9,1X,I4)
  7 FORMAT(I4,1X,I4,1X,I4,1X,I4,1X,I4,1X,I4,1X,I4)
950 STOP
   END
```

MSORT SAS

```
* THIS PROGRAM SORTS THE MIX RECORDS BY BOD;
CMS FILEDEF FONE DISK TOTAL MIX A;
CMS FILEDEF FTWO DISK SORT MIX A(RECFM F
    LRECL 80 BLOCK 80;
DATA DONE;
   INFILE FONE;
   INPUT E2NUM $ 1-4 ASRNUM $ 6-9 MOS $ 11-14 GD $ 16
         MCC $ 18-20 AMT $ 22-24 SPL $ 26
         BOD $ 28-36 BMIS $ 38-41;
PROC SORT;
   BY BOD;
DATA _NULL_;
   SET DONE;
   FILE FTWO;
   PUT E2NUM 1-4 ASRNUM 6-9 MOS 11-14 GD 16 MCC 18-20
       AMT 22-24 SPL 26 BOD 28-36 BMIS 38-41;
```

```
* PROGRAM TO MERGE E1 AND E2ASR MIX FILES
      PROGRAM DEMAND
      CHARACTER*1 UR,SEX,EXP
      CHARACTER*2 CGP,CGPA(20),PGP,PGPA(20)
      CHARACTER*3 MCC
      CHARACTER*8 MAC,OT,OTA(20)
      CHARACTER*9 BODM,BODE,TBOD
      INTEGER MMOS,PRIA(20)
      INTEGER EMOS,AMOS1,AMOS2,PMA(20),AM1(20),AM2(20)
      INTEGER E2NUM,ASRNUM,E1NUM,E1NA(20),AMT,SPL,PRI
      INTEGER MATCH,TCNT,I,BMIS,JMOS,JM(20),GD,GDA(20),F
      OPEN (UNIT = 1, FILE = 'GROUPE1 FILE A', STATUS = 'OLD')
      OPEN (UNIT = 2, FILE = 'SORT MIX A', STATUS = 'OLD')
      OPEN (UNIT = 7, FILE = 'SPL2 DEMAND A', STATUS = 'NEW')
      OPEN (UNIT = 8, FILE = 'SPL3 DEMAND A', STATUS = 'NEW')
      OPEN (UNIT = 9, FILE = 'SPL4 DEMAND A', STATUS = 'NEW')
      OPEN (UNIT = 10, FILE = 'SPL5 DEMAND A', STATUS = 'NEW')
      MATCH = 0
      TCNT = 1
* READ IN SORT MIX VALUES
      READ (2,2,END = 200) E2NUM,ASRNUM,MCC,AMT,SPL,
     C                     BODM,BMIS
      TBOD = BODM
      IF (SPL .EQ. 2) THEN
         F = 7
      ELSE IF (SPL .EQ. 3) THEN
         F = 8
      ELSE IF (SPL .EQ. 4) THEN
         F = 9
      ELSE IF (SPL .EQ. 5) THEN
         F = 10
      END IF
* READ IN E1 BOD AND COMPARE WITH SORT MIX BOD
   20 READ (1,3,END = 200) E1NUM,BODE,PRI,GD,EMOS,OT,
     C                     AMOS1,AMOS2,JMOS,CGP,PGP
   25 IF (BODM .EQ. BODE) THEN
         MATCH = 1
         PRIA(TCNT) = PRI
         GDA(TCNT) = GD
         PMA(TCNT) = EMOS
         AM1(TCNT) = AMOS1
         AM2(TCNT) = AMOS2
         JM(TCNT) = JMOS
         CGPA(TCNT) = CGP
         PGPA(TCNT) = PGP
         OTA(TCNT) = OT
         E1NA(TCNT) = E1NUM
         TCNT = TCNT + 1
         GOTO 20
      ELSE
         IF (MATCH .EQ. 1) THEN
            DO 30 I=1,TCNT-1
              WRITE (F,4) PMA(I),GDA(I),MCC,AM1(I),AM2(I),
     C              PRIA(I),OTA(I),SPL,AMT,JM(I),BMIS,
     C              CGPA(I),PGPA(I),E1NA(I),E2NUM,ASRNUM
   30       CONTINUE
* READ IN SORT MIX DATA AND CHECK FOR MATCH
* WITH PREVIOUS SORT MIX DATA
   35       READ (2,2,END = 200) E2NUM,ASRNUM,MCC,AMT,SPL,
     C                     BODM,BMIS
      IF (SPL .EQ. 2) THEN
         F = 7
      ELSE IF (SPL .EQ. 3) THEN
         F = 8
      ELSE IF (SPL .EQ. 4) THEN
         F = 9
      ELSE IF (SPL .EQ. 5) THEN
         F = 10
```

```
    END IF
          IF (BODM .EQ. TBOD) THEN
            DO 40 I = 1,TCNT-1
              WRITE(F,4) PMA(I),GDA(I),MCC,AM1(I),AM2(I),
     C                   PRIA(I),OTA(I),SPL,AMT,JM(I),BMIS,
     C                   CGPA(I),PGPA(I),E1NA(I),E2NUM,ASRNUM

 40           CONTINUE
              GOTO 35
          ELSE
              TCNT = 1
              MATCH = 0
              TBOD = BODM
              GOTO 25
          END IF
       ELSE
          GOTO 20
       END IF
    END IF
  3 FORMAT (I4,1X,A9,1X,I1,1X,I1,3X,I4,1X,A8,1X,I4,1X,
     C        I4,1X,I4,14X,A2,1X,A2)
  2 FORMAT (I4,1X,I4,8X,A3,1X,I3,1X,I1,1X,A9,1X,I4)
  4 FORMAT (I4,1X,I1,1X,A3,1X,I4,1X,I4,1X,I1,1X,A8,1X,
     C        I1,1X,I3,1X,I4,1X,I4,1X,A2,1X,A2,1X,I4,1X,
     C        I4,1X,I4
200 STOP
    END
```

```
C234567
C THIS PROGRAM MAKES THE MATCHES FOR THE MOVERS
      PROGRAM MMATCH
      CHARACTER*3 PMCC,BMCC,CPM(8000),CPMA(4000)
      CHARACTER*3 CPMC(4000),CPMB(2000),CTEMP(8000)
      CHARACTER*8 OT
      INTEGER OTP(1:8,0:20),PTR(0:232,0:6,0:8,1:2)
      INTEGER SPL,PRI,INVNUM,BMIS,CGP,PGP
      INTEGER PM(1:8000,1:8),PMA(1:4000,1:8),PMB(1:2000,1:8)
      INTEGER PTRA(0:232,0:6,0:8,1:2), PTRB(0:232,0:6,0:8,1:2)
      INTEGER M1,MA1,MA2,TMOS,TGD,TCGP,AMOS1,AMOS2,JMOS
      INTEGER CDATA(0:26,0:8),J,K,L,M,N,TEMP(1:4000,1:3)
      INTEGER BMOS,BGD,BAM1,BAM2,AMT,BJMOS,BCGP
      INTEGER PTRC(0:232,0:6,0:8,1:2),PMC(1:4000,1:8)
      INTEGER MOS,MOSNUM,OCFC,SOCFC,OCMOS,SOCMOS,TNUM
      INTEGER OCC(0:99,1:2),SOCC(0:999,1:2),TOCC,TSOCC,I
      INTEGER R,Q,QMOS,TUF,PMOS,GD
      DATA PM,PMA,PMB / 64000*0,32000*0,16000*0 /
      DATA PTR,PTRA,PTRB / 29358*0,29358*0,29358*0 /
      DATA TMOS,TGD,TCGP / 0, 0, 0 /
      DATA PTRC,PMC / 29358*0, 32000*0 /
      DATA OCC,SOCC / 200*0, 2000*0 /
      DATA TOCC,TSOCC,TNUM,TSNUM / 0, 0, 0, 0 /
      DATA OTP / 168*0 /
      OPEN (UNIT = 1, FILE = 'MG FILE A', STATUS = 'OLD')
      OPEN (UNIT = 2, FILE = 'MG PAMOS1 A', STATUS = 'OLD')
      OPEN (UNIT = 3, FILE = 'MG PAMOS2 A', STATUS = 'OLD')
      OPEN (UNIT = 4, FILE = 'UNSORTED MATCH A', STATUS = 'NEW')
      OPEN (UNIT = 7, FILE = 'SPL2 DEMAND A', STATUS = 'OLD')
      OPEN (UNIT = 8, FILE = 'CGROUP DATA A', STATUS = 'OLD')
      OPEN (UNIT = 9, FILE = 'OT PTR A', STATUS = 'OLD')
      OPEN (UNIT = 10, FILE = 'AMOS MG A', STATUS = 'OLD')
C READ IN THE CGP MAPPING DATA INTO CDATA ARRAY
      DO 5 I = 0,26
         READ(8,13) CDATA(I,0),CDATA(I,1),CDATA(I,2),CDATA(I,3),
     C              CDATA(I,4),CDATA(I,5),CDATA(I,6),CDATA(I,7),
     C              CDATA(I,8)
    5 CONTINUE
C     WRITE(6,*) 'CDATA(24,1) = ',CDATA(24,1)
C THIS ROUTINE CREATES POINTERS INTO THE PMOS ARRAY FOR OCC FIELD AND
C SUBOCC FIELD MOS'S
      OCFC = 100
      SOCFC = 10
    6 READ (8,15,END = 7) MOS,MOSNUM
      OCMOS = MOS/OCFC
      SOCMOS = MOS/SOCFC
      IF (OCMOS .NE. TOCC) THEN
         OCC(OCMOS,1) = MOSNUM
         OCC(TOCC,2) = TNUM
         TOCC = OCMOS
      END IF
      TNUM = MOSNUM
      IF (SOCMOS .NE. TSOCC) THEN
         SOCC(SOCMOS,1) = MOSNUM
         SOCC(TSOCC,2) = TSNUM
         TSOCC = SOCMOS
      END IF
      TSNUM = MOSNUM
      GO TO 6
    7 OCC(TOCC,2) = TNUM
      SOCC(TSOCC,2) = TSNUM
C     WRITE(6,*) 'TNUM = ',TNUM
C READ IN THE OFFICER TYPE MOS'S
      IF (1 .EQ. 1) GOTO 950
      DO 9 I = 1,8
         READ(9,16) OTP(I,0),OTP(I,1),OTP(I,2),OTP(I,3),
     C              OTP(I,4),OTP(I,5),OTP(I,6),OTP(I,7),
     C              OTP(I,8),OTP(I,9),OTP(I,10),OTP(I,11),
```

```
     C              OTP(I,12),OTP(I,13),OTP(I,14),OTP(I,15)
     9 CONTINUE
C      WRITE(6,*) 'OTP(8,2) = ',OTP(8,2)
C READ IN PMOS MOVERS AND SET UP ARRAYS
   950 I = 1
       TMOS = 0
       TGD = 0
       TCGP = 0
    10 READ(1,11,END = 19) PMOS,GD,AMOS1,AMOS2,PMCC,JMOS,INVNUM,CGP
       IF (PMOS .NE. TMOS .OR. GD .NE. TGD .OR. CGP .NE. TCGP) THEN
          PTR(PMOS,GD,CGP,1) = I
          PTR(TMOS,TGD,TCGP,2) = I-1
       END IF
       PM(I,1) = INVNUM
       PM(I,2) = PMOS
       PM(I,3) = GD
       PM(I,4) = CGP
       CPM(I) = PMCC
       PM(I,6) = AMOS1
       PM(I,7) = AMOS2
       PM(I,8) = JMOS
       TMOS = PMOS
       TGD = GD
       TCGP = CGP
       I = I + 1
       GO TO 10
    19 M1 = I - 1
C      WRITE(6,*) 'M1 =  ',M1
C READ IN PAMOS1 MOVERS AND SET UP ARRAY
       I = 1
       TMOS = 0
       TGD = 0
       TCGP = 0
    20 READ(2,11,END = 25) PMOS,GD,AMOS1,AMOS2,PMCC,JMOS,INVNUM,CGP
       IF (PMOS .NE. TMOS .OR. GD .NE. TGD .OR. CGP .NE. TCGP) THEN
          PTRA(PMOS,GD,CGP,1) = I
          PTRA(TMOS,TGD,TCGP,2) = I-1
       END IF
       PMA(I,1) = INVNUM
       PMA(I,2) = PMOS
       PMA(I,3) = GD
       PMA(I,4) = CGP
       CPMA(I) = PMCC
       PMA(I,6) = AMOS1
       PMA(I,7) = AMOS2
       PMA(I,8) = JMOS
       TMOS = PMOS
       TGD = GD
       TCGP = CGP
       I = I + 1
       GO TO 20
    25 MA1 = I - 1
C READ IN PAMOS2 MOVERS AND SET UP ARRAY
       I = 1
       TMOS = 0
       TGD = 0
       TCGP = 0
    30 READ(3,11,END = 35) PMOS,GD,AMOS1,AMOS2,PMCC,JMOS,INVNUM,CGP
       IF (PMOS .NE. TMOS .OR. GD .NE. TGD .OR. CGP .NE. TCGP) THEN
          PTRB(PMOS,GD,CGP,1) = I
          PTRB(TMOS,TGD,TCGP,2) = I-1
       END IF
       PMB(I,1) = INVNUM
       PMB(I,2) = PMOS
       PMB(I,3) = GD
       PMB(I,4) = CGP
       CPMB(I) = PMCC
       PMB(I,6) = AMOS1
       PMB(I,7) = AMOS2
       PMB(I,8) = JMOS
```

```
      TMOS = PMOS
      TGD = GD
      TCGP = CGP
      I = I + 1
      GO TO 30
   35 MA2 = I - 1
C     WRITE(6,*) 'MA2 = ',MA2
C READ IN PAMOS1 MOVERS IN AMOS ORDER AND SET UP ARRAY
      I = 1
      TMOS = 0
      TGD = 0
      TCGP = 0
   40 READ(10,11,END = 45) PMOS,GD,AMOS1,AMOS2,PMCC,JMOS,INVNUM,CGP
      IF (PMOS .NE. TMOS .OR. GD .NE. TGD .OR. CGP .NE. TCGP) THEN
         PTRC(AMOS1,GD,CGP,1) = I
         PTRC(TMOS,TGD,TCGP,2) = I-1
      END IF
      PMC(I,1) = INVNUM
      PMC(I,2) = PMOS
      PMC(I,3) = GD
      PMC(I,4) = CGP
      CPMC(I) = PMCC
      PMC(I,6) = AMOS1
      PMC(I,7) = AMOS2
      PMC(I,8) = JMOS
      TMOS = AMOS1
      TGD = GD
      TCGP = CGP
      I = I + 1
      GO TO 40
   45 MA3 = I - 1
      WRITE(6,*) 'MA3 = ',MA3
C READ IN DEMANDS AND MAKE MATCHES
   50 READ(7,12,END = 200) BMOS,BGD,BMCC,BAM1,BAM2,PRI,OT,SPL,AMT,
     C                     BJMOS,BMIS,BCGP,PGP
      IF (PGP .EQ. 1) GOTO 60
      IF (PGP .EQ. 2) GOTO 70
      IF (PGP .EQ. 3) GOTO 80
      IF (PGP .EQ. 4) GOTO 90
      IF (PGP .EQ. 5) GOTO 50
      IF (PGP .EQ. 6) GOTO 110
C PMOS AND 2 AMOS - USE PTRB
   60 N = 0
      DO 62 J = 1,CDATA(BCGP,0)
         L = CDATA(BCGP,J)
         IF (PTRB(BMOS,BGD,L,1) .NE. 0) THEN
            DO 64 K = PTRB(BMOS,BGD,L,1),PTRB(BMOS,BGD,L,2)
               IF (PMB(K,6) .EQ. BAM1 .AND. PMB(K,7) .EQ. BAM2) THEN
                  N = N + 1
                  TEMP(N,1) = PMB(K,1)
                  TEMP(N,2) = PMB(K,2)
                  CTEMP(N) = CPMB(K)
               END IF
   64       CONTINUE
         END IF
   62 CONTINUE
      IF (N .GT. 0) THEN
         DO 66 I = 1,N
            WRITE(4,14) TEMP(I,1), TEMP(I,2),CTEMP(I),
     C                  BMCC,BMIS,PRI,AMT
   66    CONTINUE
      END IF
      GO TO 50
C PMOS AND 1 AMOS - USE PTRA
   70 N = 0
      DO 72 J = 1,CDATA(BCGP,0)
         L = CDATA(BCGP,J)
C        WRITE(6,*) 'PMOS AND AMOS'
         IF (PTRA(BMOS,BGD,L,1) .NE. 0) THEN
            DO 74 K = PTRA(BMOS,BGD,L,1),PTRA(BMOS,BGD,L,2)
```

```
                IF (PMA(K,6) .EQ. BAM1 .OR. PMA(K,7) .EQ. BAM1) THEN
                   N = N + 1
                   TEMP(N,1) = PMA(K,1)
                   TEMP(N,2) = PMA(K,3)
                   CTEMP(N) = CPMA(K)
                END IF
   74        CONTINUE
          END IF
   72  CONTINUE
       IF (N .GT. 0) THEN
          DO 76 I = 1,N
             WRITE(4,14) TEMP(I,1), TEMP(I,2),CTEMP(I),
     C                   BMCC,BMIS,PRI,AMT
   76     CONTINUE
       END IF
       GO TO 50
C PMOS ONLY - USE PTR
   80  N = 0
       DO 82 J = 1,CDATA(BCGP,0)
          L = CDATA(BCGP,J)
          IF (PTR(BMOS,BGD,L,1) .NE. 0) THEN
             DO 84 K = PTR(BMOS,BGD,L,1),PTR(BMOS,BGD,L,2)
                   N = N + 1
                   TEMP(N,1) = PM(K,1)
                   TEMP(N,2) = PM(K,2)
                   CTEMP(N) = CPM(K)
   84        CONTINUE
          END IF
   82  CONTINUE
       IF (N .GT. 0) THEN
          DO 86 I = 1,N
             WRITE(4,14) TEMP(I,1), TEMP(I,2), CTEMP(I),
     C                   BMCC,BMIS,PRI,AMT
   86     CONTINUE
       END IF
       GO TO 50
C OCC FIELD OR SUBOCC FIELD
C HAVE SPEARATE PARTS FOR OCC FIELD AND SUBOCC
C HAVE ANOTHER LOOP TO LOOP THROUGH THE MOS'S
   90  N = 0
       DO 92 J = 1,CDATA(BCGP,0)
          L = CDATA(BCGP,J)
          TUF = BMOS - 1000
          IF (TUF .GE. 100) THEN
            DO 95 BMOS = SOCC(TUF,1),SOCC(TUF,2)
            IF (PTR(BMOS,BGD,L,1) .NE. 0) THEN
               DO 94 K = PTR(BMOS,BGD,L,1),PTR(BMOS,BGD,L,2)
                     N = N + 1
                     TEMP(N,1) = PM(K,1)
                     TEMP(N,2) = PM(K,2)
                     CTEMP(N) = CPM(K)
   94          CONTINUE
            END IF
   95       CONTINUE
          ELSE
            DO 97 BMOS = OCC(TUF,1),SOCC(TUF,2)
            IF (PTR(BMOS,BGD,L,1) .NE. 0) THEN
               DO 96 K = PTR(BMOS,BGD,L,1),PTR(BMOS,BGD,L,2)
                     N = N + 1
                     TEMP(N,1) = PM(K,1)
                     TEMP(N,2) = PM(K,2)
                     CTEMP(N) = CPM(K)
   96          CONTINUE
            END IF
   97       CONTINUE
          END IF
   92  CONTINUE
       IF (N .GT. 0) THEN
          DO 98 I = 1,N
             WRITE(4,14) TEMP(I,1), TEMP(I,2), CTEMP(I),
```

```
     C                    BMCC,BMIS,PRI,AMT
    98    CONTINUE
          END IF
          GO TO 50
C OFFICER TYPE
  100 N = 0
          DO 102 J = 1,CDATA(BCGP,0)
             L = CDATA(BCGP,J)
             DO 105 Q = 1,8
                IF (OT(Q:Q) .EQ. '*') THEN
                  DO 106 R = 1,OTP(Q,0)
                     QMOS = OTP(Q,R)
                     IF (PTR(QMOS,BGD,L,1) .NE. 0) THEN
                       DO 104 K = PTR(QMOS,BGD,L,1),PTR(QMOS,BGD,L,2)
                          N = N + 1
                          TEMP(N,1) = PM(K,1)
                          TEMP(N,2) = PM(K,2)
                          CTEMP(N) = CPM(K)
  104                   CONTINUE
                     END IF
  106              CONTINUE
                 END IF
  105          CONTINUE
  102 CONTINUE
          IF (N .GT. 0) THEN
             DO 107 I = 1,N
             WRITE(4,14) TEMP(I,1), TEMP(I,2), CTEMP(I),
     C                    BMCC,BMIS,PRI,AMT
  107    CONTINUE
          END IF
          GO TO 50
C AMOS ONLY
C SET UP ANOTHER INDEX ON THE AMOS1 FILE IN AMOS ORDER
  110 N = 0
          DO 112 J = 1,CDATA(BCGP,0)
             L = CDATA(BCGP,J)
             IF (PTRC(BAM1,BGD,L,1) .NE. 0) THEN
                DO 114 K = PTRC(BAM1,BGD,L,1),PTRC(BAM1,BGD,L,2)
                   N = N + 1
                   TEMP(N,1) = PMC(K,1)
                   TEMP(N,2) = PMC(K,2)
                   CTEMP(N) = CPMC(K)
  114          CONTINUE
             END IF
  112 CONTINUE
          IF (N .GT. 0) THEN
             DO 116 I = 1,N
             WRITE(4,14) TEMP(I,1), TEMP(I,2), CTEMP(I),
     C                    BMCC,BMIS,PRI,AMT
  116    CONTINUE
          END IF
          GO TO 50
    11 FORMAT (I4,1X,I1,1X,I4,1X,I4,3X,A3,5X,I4,1X,I5,1X,I1)
    12 FORMAT (I4,1X,I1,1X,A3,1X,I4,1X,I4,1X,I1,1X,A8,1X,I1,1X,
     C         I3,1X,I4,1X,I4,1X,I2,1X,I2)
    13 FORMAT (I1,1X,I1,1X,I1,1X,I1,1X,I1,1X,I1,1X,I1,1X,I1,1X,I1)
    14 FORMAT (I5,1X,I1,1X,A3,1X,A3,1X,I4,1X,I1,1X,I3)
    15 FORMAT (I4,1X,I3)
    16 FORMAT (I4,15(1X,I4))
   200 STOP
          END
```

APPENDIX E
FINAL PROCESSING PROGRAMS

SHUFFLE SAS

```
c this program sorts the match data by bmis and then;
c by invnum for use by the solver;
CMS FILEDEF FONE DISK UNSORTED MATCH A;
CMS FILEDEF FTWO DISK BMIS MATCH A(RECFM F
    lrecl 80 block 80;
DATA DONE;
  INFILE FONE;
  INPUT INVNUM 1-5 GD $ 7 PMCC $ 9-11 BMCC $ 13-15
        bmis 17-20 PRI $ 22 SPL $ 24-26;
PROC SORT;
  BY BMIS INVNUM;
DATA _NULL_;
  SET DONE;
  FILE FTWO;
  PUT INVNUM 1-5 GD $ 7 PMCC $ 9-11 BMCC $ 13-15
      bmis 17-20 PRI $ 22 SPL $ 24-26;
```

```
      PROGRAM GRUN
      INTEGER CAPSUM,TINV,INVNOD(21100),TBMIS
      INTEGER BMIS,INVNUM,PRI,AMT,TEMP
      CHARACTER*1 GD
      CHARACTER*3 PMCC,BMCC
      INTEGER IOD,IAD,M
      INTEGER IER,ISCALE,IPVT
      INTEGER NODSUM,MP1,N,IND,IHS,IBIG
      INTEGER MAXC,ISUP,MBIG,IPTG,IOUT
      INTEGER NNE,NNS,IPG,NAP,IWU,I
      INTEGER*2 T,P,D,IT,NSA,A
      INTEGER*4 H,C,CP,X,CPX,U,ISA,COST
      DIMENSION H(9000),X(9000),T(9000),C(9000)
      DIMENSION CP(9000),CPX(9000),P(9000)
      DIMENSION D(9000),IT(9000),U(9000)
      DIMENSION NSA(9000),ISA(9000),A(9000)
      DATA H,X,T,C,CP / 9000*0, 9000*0, 9000*0, 9000*0 , 9000*0 /
      DATA INVNOD / 21100*0 /
C PUT IN THE INVENTORY NODE DATA
      WRITE(6,*) 'GETTING INVENTORY DATA'
      CAPSUM = 0
      TINV = 0
      NODSUM = 1
   10 READ(1,2,END = 15) INVNUM
      IF (INVNUM .NE. TINV) THEN
         H(NODSUM) = 1
         X(NODSUM) = 1
         WRITE(8,1) H(NODSUM),X(NODSUM)
         CAPSUM = CAPSUM + 1
         INVNOD(INVNUM) = NODSUM
         NODSUM = NODSUM + 1
      END IF
         TINV = INVNUM
         GOTO 10
C I MAY LATER NEED TO ADD ANOTHER SUPPLY NODE HERE TO HELP
C MAKE THE SUPPLY AND DEMAND EQUAL
   15 TEMP = NODSUM
      WRITE(6,*) 'NODSUM   CAPSUM'
      WRITE(6,*) NODSUM,CAPSUM
      NODSUM = NODSUM + 1
C READ IN AND SET BMIS NODE DATA
      WRITE (6,*) 'GETTING BMIS DATA'
      N = 1
      TBMIS = 0
   20 READ(2,3, END = 50) INVNUM,GD,PMCC,BMCC,BMIS,PRI,AMT
      IF (BMIS .NE. TBMIS) THEN
         H(NODSUM) = N
         X(NODSUM) = 0 - AMT
         WRITE(8,1) H(NODSUM),X(NODSUM)
         CAPSUM = CAPSUM - AMT
         NODSUM = NODSUM + 1
      END IF
      T(N) = INVNOD(INVNUM)
      C(N) = PRI
      CP(N) = 1
      WRITE(9,4) T(N),C(N),CP(N)
      N = N + 1
      TBMIS = BMIS
      GOTO 20
C MAKE SUPPLY AND DEMAND EQUAL
   50 WRITE(6,*) 'NODSUM   CAPSUM    N'
      WRITE(6,*) NODSUM,CAPSUM,N
      IF (CAPSUM .LT. 0) THEN
         H(TEMP) = 1
         X(TEMP) = 1 - CAPSUM
         H(NODSUM) = N
         X(NODSUM) = -1
         T(N) = TEMP
```

```
            C(N) = 0
            CP(N) = 0 - CAPSUM
            H(NODSUM + 1) = N + 1
            M = NODSUM - 1
            N = N - 1
         ELSE IF (CAPSUM .GT. 0) THEN
            H(TEMP) = 1
            X(TEMP) = 1
            H(NODSUM) = N
            X(NODSUM) = 0 - CAPSUM
            T(N) = TEMP
            C(N) = 0
            CP(N) = CAPSUM
            H(NODSUM + 1) = N + 1
            M = NODSUM - 1
            N = N - 1
         ELSE
            H(NODSUM) = N
            M = NODSUM - 1
            N = N - 1
         END IF
C SET ALL OF THE STANDARD VALUES FOR GNET
         MP1 = M + 1
         IND = MP1
         IAD = N
         IQD = IND
         IHS = 0
         IBIG = 1000000000
         MAXC = 0
         ISUP = -1
         MBIG = -1
         IOUT = 4
         NNE = -1
         NNS = -1
         IPG = -1
         NAP = -1
         IPTG = 3
         WRITE(6,*) 'CALLING GNET'
C CALL GNET
C        CALL GNETX(IND,IAD,IQD,M,H,T,C,CP,X,CPX,P,D,IT,U,NSA,ISA,A,IHS,
C       1 IBIG,MAXC,ISUP,MBIG,NNE,NNS,IPG,NAP,IPTG,IOUT,IER,ISCALE,IPVT)
C         WRITE(8,59)
C   59 FORMAT( ' I,X(I),P(I)' )
C        DO 60 I = 1,N
C           WRITE(8,8) I,X(I),CPX(I),P(I)
C   60 CONTINUE
      1 FORMAT(I4,2X,I3)
      2 FORMAT(I5)
      3 FORMAT(I5,1X,A1,1X,A3,1X,A3,1X,I4,1X,I1,1X,I3)
      4 FORMAT(I5,2X,I3,2X,I3)
      8 FORMAT(1X, I2, I4,2X,I4)
    200 STOP
         END
```

LIST OF REFERENCES

1. Exner, P. J., *A Prototype Decision Support System for Marine Corps Officer Allocation Policy Analysis*, Master's Thesis, Naval Postgraduate School, Monterey, CA, Sept. 1987.

2. Klingman, D., Mead, M., and Phillips, N. V., "Network Optimization Models for Military Manpower Planning," *Operational Research* pp. 786-800, 1984.

3. Klingman, D., Mead, M., and Phillips, N. V., "Topological and Computational Aspects of Preemptive Multicriteria Military Personnel Assignment Problems", *Management Science* v. 30, No. 11, pp. 1362-1375, Nov. 1984.

4. Rosenthal, R. E., "Principles of Multiobjective Optimization," *Decision Sciences* v16, No.2, pp 133-152, Spring, 1985.

5. Bradley, G., Brown, G., and Graves, G.,"Design and Implementation of Large Scale Primal Transshipment Algorithms," *Management Science* v. 24, No.1, pp.1-34, Sept. 1977.

6. *Officer Staffing Goal Model (OSGM) Users Guide*, Decision Systems Associates, Inc., Rockville, MD, September 1983.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Department Chairman, Code 52 Computer Science Department Naval Postgraduate School Monterey, CA 93943	2
4.	Professor Gordon H. Bradley Code 52BZ Computer Science Department Naval Postgraduate School Monterey, CA 93943	4
5.	Associate Professor C. Thomas Wu Code 52WQ Computer Science Department Naval Postgraduate School Monterey, CA 93943	1
6.	Major D. Hundley Code MMOA-3 Headquarters United States Marine Corps Washington, DC 20380-0001	2
7.	Captain Mark E. Hayes 134 Cresston Road Arnold,MD 21012	4


Thesis
H4048
c.1
Hayes
Design for a prototype
Marine Corps officer
staffing model.
Thesis
H4048
c.1
Hayes
Design for a prototype
Marine Corps officer
staffing model.